Departamento de Informática

PROYECTO FIN DE CARRERA

UM4NI: User Models for Natural Interaction

Autor: Leonardo Castaño Zabaleta

Tutor: Francisco Javier Calle Gómez

Director: Elena Castro Galán,

Leganés, julio de 2010

Título: User Models for Natural Interaction

Autor: Leonardo Castaño Zabaleta

Tutor: Francisco Javier Calle Gómez

Director: Elena Castro Galán

EL TRIBUNAL

PRESIDENTE: Manuel Velasco de Diego

VOCAL: Dolores Cuadra Fernández

SECRETARIO: Jessica Rivero Espinosa

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

Agradecimientos

En primer lugar agradecer a toda mi familia por aguantarme durante la realización de este proyecto y por apoyarme en los días buenos y en los malos. En segundo lugar agradecer especialmente a Javier y a Elena que estos últimos meses prácticamente se convirtieron en mi segunda familia y supieron guiarme y ayudarme durante todas las fases del proyecto. Sin duda alguna este proyecto ni hubiese tenido cabida ni se hubiese podido llevar a cado sin su ayuda, colaboración y disposición.

También me gustaría agradecer a todos los miembros del grupo de Bases de Datos Avanzadas que me ayudaron en la realización de este proyecto y en mi formación y sin los cuales este proyecto no hubiese sido posible.

Finalmente agradecer también a Pablo y Julián por las horas dedicadas y por la ayuda que recibí de su parte en momentos que quizás los inexplicables misterios de Oracle superaban a toda lógica y razón.

A todos ellos muchísimas gracias.

Resumen

El proyecto que nos ocupa se centra en el campo de la interacción natural y más concretamente en el campo de los modelos de usuario. Se trata de llevar a cabo un proyecto de investigación en el ámbito del modelado de usuarios. A lo largo del proyecto se llevaron a cabo dos modelos de usuario con distinto tratamiento de valores. En cuanto a las características de dichos modelos de usuario, se trata de modelos de usuario "automodelados", es decir, modelos basados en la experiencia y no en estereotipos o perfiles como los existentes hasta el momento. Por lo tanto, el proyecto actual es una innovación en el campo de los modelos de usuario y de la interacción natural.

El proyecto se descompuso en varias fases entre las que se deben destacar una fase de planteamiento de objetivos donde se identificaron las metas del proyecto, una fase de análisis en la que se analizó el problema y las alternativas o soluciones, una fase de diseño, una fase de implementación y por último una validación y evaluación del sistema.

En lo que se refiere a la evaluación del modelo, esta se realizó con un conjunto de datos reales. Se construyó una batería de experimentos con la que probar el funcionamiento de los modelos. En primer lugar destacar que el dominio en el que se probó el modelo es un dominio muy abierto y difuso en el que la tasa de acierto para los valores por defecto es inferior al 1%. En todo caso los modelos desarrollados en el proyecto obtuvieron tasas de acierto no inferiores a un 10% y alcanzaron un máximo de un 13%. Por lo que los resultados obtenidos teniendo en cuenta la morfología del dominio son muy satisfactorios.

Palabras clave

automodelado, modelo de grupos, modelo de usuarios, certeza, probabilidad, usuario actual, contexto, base de conocimiento, encaje, fusión.

Índice

Índice figuras y tablas

1. Introducción

En los últimos años se han producido numerosos avances en el campo de los modelos de usuario y de la interacción natural. Si nos remontamos a los comienzos de esta disciplina, solamente existían unos trescientos investigadores que mediante la ayuda de las universidades trataban de explorar un campo completamente nuevo. Sin embargo, en la actualidad el modelado de usuarios como rama de investigación cuenta con su propia conferencia bianual y de la misma forma, el modelado de usuarios como tecnología se ha convertido en un estándar para muchas tareas de investigación. En años más recientes, el modelado de usuarios, se convirtió en un estándar de relevancia en sistemas de ámbito comercial.

En la actualidad las áreas de aplicación del modelado de usuarios, son bien distintas y entre ellas podemos destacar, sistemas educacionales, sistemas e-commerce e interfaces de acceso universal. En definitiva todo un conjunto de aplicaciones del modelado de usuarios, que se han desarrollado bajo el termino de personalización.

1.1. Modelos de usuario e interacción natural

Todo proceso de investigación, se encuadra principalmente en uno de los dos siguientes escenarios, un marco bueno para la creación de nuevos sistemas o tecnologías o un marco bueno para la escritura de documentos sobre los nuevos sistemas o tecnologías. La historia de la interacción y de las interfaces, se puede entender como un largo camino desde la complejidad de las maquinas diseñadas con propósitos científicos hasta los dispositivos masivos de uso personal. Lo que supone un marco perfecto para la investigación en los dos ámbitos mencionados anteriormente. En la actualidad la mayoría de los prototipos y sistemas se desarrollan atendiendo a los principios de personalización y por tanto acomodándolos a las diferentes necesidades de los usuarios.

Como ya se comento anteriormente, la interacción hombre maquina se realiza sin que en muchos casos la maquina sea capaz de tener conocimiento del estado emocional del humano o de su situación social por ejemplo. Los modelos de usuario tratan de acercar estas dos posiciones permitiéndole a la maquina inferir conocimiento sobre el humano que esta a priori desconoce.

1.2. Conceptos básicos

En la presente sección, se detallan algunos de los conceptos básicos a cerca del modelado de usuarios y de la interacción natural, que ayudaran al lector en la comprensión de las distintas secciones de este documento.

- Certeza: Calidad de cierto.
- Certidumbre: Conocimiento seguro y claro de algo.
- Circunstancia: Conjunto de características en que se desarrolla la interacción.
- Precisión: Medida de exactitud o fidelidad. Proporción de material recuperado realmente relevante.
- Probabilidad: Frecuencia con que se obtiene un resultado al llevar a cabo un experimento aleatorio.
- Recall: Medida de completitud. Proporción de material relevante recuperado.
- Usuario actual o contexto de usuario: Conjunto de características conocidas del interlocutor.
- Veracidad: Grado de cumplimiento con la realidad de la información. Conformidad con la verdad.

1.3. Modelos de usuario de grupos y de perfiles

Desde los comienzos del paradigma de modelado de usuario, la mayoría de las herramientas desarrolladas bajo la idea de personalización, se centran en crear modelos de usuario basados en perfiles o bien modelos de usuario orientados a un dominio concreto. A diferencia de la mayoría de estas herramientas, la alternativa que aquí se propone se basa en un modelo de usuario de grupos basado en la experiencia, es decir un modelo completamente independiente de la aplicación y que permitirá a esta adaptarse a los distintos usuarios. Los términos de personalización y adaptabilidad provienen de la heterogeneidad de usuarios existentes, esta heterogeneidad interviene en el diseño de cualquier sistema interactivo. Así pues el diseño de sistemas capaces de adaptarse a los usuarios enfatiza conceptos como la usabilidad y la utilidad de los sistemas.

Según se vio anteriormente, existen varios enfoques para el desarrollo de modelos de usuario, los modelos de usuario basados en perfiles o estereotipos, estos modelos establecen estereotipos de usuarios basados en características comunes. Dichos estereotipos están prefijados inicialmente, es decir antes de que se produzca la interacción hombre maquina, lo cual supone que durante el proceso de interacción, el usuario percibirá respuestas relativas al

perfil con el que mejor clasifica. Este tipo de modelos tuvieron su auge en el pasado y actualmente se siguen empleando cuando el dominio y tipo de usuarios que interaccionan con el sistema se puede acotar y se conoce de inicio. Por el contrario si se desconoce el dominio o si los usuarios son desconocidos o suficientemente parecidos como para que no se puedan definir características diferenciadoras a priori, estos modelos no pueden considerarse como una alternativa eficaz al problema del modelado de usuarios.

Por otro lado los modelos de usuarios de grupos, no establecen ningún estereotipo, sino que mediante el proceso de modelado de usuarios (interacción), infieren la información de usuario y crean grupos de usuarios similares, se trata por tanto de modelos basados en la experiencia y no en perfiles establecidos. Este último tipo de modelos a diferencia de los modelos basados en perfiles, no suelen estar orientados a un dominio concreto, sino que independientemente de la aplicación, son capaces de adaptarse al dominio y a los usuarios. Es por esto que a este tipo de modelos de usuario se les conoce como modelos “auto modelados”, porque el propio modelo a partir de la interacción, es capaz de crear categorías de usuarios y posteriormente clasificar cada usuario a una de estas categorías.

Este tipo de modelos de usuario cuentan con ciertas ventajas frente a los modelos de usuario basados en perfiles o estereotipos, en primer lugar los modelos “auto modelados” permiten aumentar el número de ranuras para grupos de usuarios con lo cual permiten una mejor clasificación de los usuarios en los grupos, en segundo lugar los modelos “auto modelados”, no establecen las categorías de usuarios a priori, lo que les permite ser independientes de la aplicación. El inconveniente de los modelos “auto modelados” frente a los basados en perfiles o estereotipos, es que del tiempo de procesamiento, deben emplear una cierta parte en tareas de autoconfiguración, reduciendo por tanto el tiempo o la eficiencia con la que se atenderá al usuario.

2. Estado del Arte

En el presente capitulo, se pretende ofrecer un resumen de los distintos tipos de modelos de usuario existentes, así como de los distintos enfoques empleados para su desarrollo.

2.1. Historia de los primeros modelos de usuario

La historia de los modelos de usuario genéricos, se remonta a la década de los setenta, donde comenzaron a surgir las primeras aproximaciones a los modelos de usuario, con los trabajos de Cohen y Perrault, 1979, Allen, 1979 o Elaine Rich también en 1979.

Durante estos primeros trabajos sobre modelado de usuarios, no se establecieron las distinciones necesarias entre componentes que ayudan al modelado de usuarios y componentes que realizan otras tareas. A partir de mediados de los años ochenta, dichas distinciones comenzaron a llevarse a cabo, sin embargo no se realizaron los esfuerzos suficientes para hacer los componentes de los modelos de usuario reutilizables en sistemas futuros. Así pues, tal y como se menciono anteriormente, las primeras aproximaciones el modelado de usuario, surgen en la década de los setenta, se trata de aproximaciones que en cierta medida logran recuperar cierta información sobre los usuarios, y son capaces de modificar ligeramente su comportamiento para adaptarse a los usuarios.

Durante estos primeros trabajos, el modelado de usuario se concebía como una aplicación en sí, y no se establecían las diferencias necesarias entre aquellos componentes que sirven al modelo de usuario y los que simplemente llevan a cabo otras tareas. A partir de mediados de los años ochenta, estas distinciones comenzaron a tenerse en cuenta, sin embargo no se realizaron aun los esfuerzos suficientes para hacer cada componente de los modelos de usuario reutilizables en futuros sistemas adaptativos.

2.2. Primeros modelos

A continuación se exponen los primeros modelos de usuario, que comenzaron a tener repercusión en la década de los ochenta.

2.2.1. GUMS

En 1986, Tim Finin [Finin, T. W., 1989] publicó su sistema genérico de modelado de usuarios GUMS (General user modelling system). Dicho software permitía definir sencillas jerarquías de estereotipos y para cada estereotipo, hechos y reglas definiendo el comportamiento

de éste. En tiempo de ejecución, GUMS es capaz de aceptar y almacenar nuevos hechos a cerca del usuario y verificar la consistencia de los nuevos hechos, con las asunciones anteriores.

GUMS sirvió de punto de partida a los siguientes modelos de usuario, ya que fue el creador de un marco que incluye servicios de modelado de usuario configurables en tiempo de ejecución, este marco fue empleado por varios modelos posteriores.

GUMS se debe clasificar dentro de los sistemas de estereotipos, dichos estereotipos se relacionan jerárquicamente, mediante una estructura arbórea. El sistema genérico de modelado de usuarios GUMS, debe ser empleado con una aplicación capaz de obtener conocimiento a cerca de los usuarios que interaccionan con ella, sin embargo la habilidad de obtener conocimiento de los usuarios en sí misma no es suficiente, ya que no se puede esperar a tener pleno conocimiento de los usuarios para comenzar a razonar sobre ellos. Afortunadamente, a menudo se pueden realizar generalizaciones de usuarios o grupos de usuarios, a dichas generalizaciones se las conoce como estereotipos. Por tanto podría definirse un estereotipo como un conjunto de reglas o hechos que aplican a un conjunto de usuarios, por tanto es posible razonar a partir de dichos estereotipos. GUMS considera hechos a toda aquella información que el sistema tiene y considera válida, por otra parte se considera una regla a aquella información que un usuario adquiere por pertenecer a un estereotipo concreto.

A la hora de almacenar nuevos hechos o nuevo conocimiento, se producen cambios en la estructura arbórea de GUMS, dichos cambios pueden dar lugar a una inconsistencia o contradicción, en este caso se requiere actualizar la estructura arbórea con el fin de encontrar una estructura que acepte el nuevo estereotipo sin producir inconsistencias, en caso de no conseguirse GUMS descarta el estereotipo.

En cuanto a las respuestas proporcionadas por GUMS, estas pueden ser de dos tipos: respuestas fuertes y respuestas débiles. El intérprete de respuestas de GUMS está escrito en Prolog, y GUMS considera respuestas fuertes aquellas que se conoce que son verdaderas o falsas, sin embargo aquellas respuestas que se “asume” que son verdaderas o falsas, pero existe desconocimiento, son consideradas respuestas débiles. El intérprete de respuestas de GUMS, muestra siempre primero las respuestas fuertes y posteriormente las débiles. Teniendo en cuenta que GUMS siempre da prioridad a las respuestas fuertes, frente a las débiles, es decir a la información fuerte frente a la débil, no necesita eliminar información débil que contradice a la información fuerte, ya que esta siempre se mostrara tras la información fuerte. GUMS emplea pesos basados en certezas, para clasificar sus respuestas en fuertes o débiles.

2.2.2.UMT

A finales de los ochenta y primeros de los noventa, grupos de investigación de distintos países, comenzaron a desarrollar nuevas estructuras y procesos dentro del marco del modelado de usuarios, con el fin de aplicar dichas estructuras y procesos en aplicaciones que se adapten a los usuarios. Es en este marco donde surge UMT (User Modelling Tool), que fue propuesto en 1994 por Brajnik, G. y Tasso, C.

UMT permite a sus desarrolladores ordenar jerárquicamente estereotipos de usuarios, así como reglas sobre dichos estereotipos, con el fin de posteriormente permitir realizar inferencia y detección de contradicciones. El sistema clasifica la información acerca de los usuarios en premisas inalterables o en suposiciones, una vez que todas las reglas de inferencia y todos los estereotipos son almacenados, el sistema busca contradicciones entre todas las suposiciones y propone varios algoritmos de resolución de contradicciones.

2.2.3.PROTUM

La herramienta PROTUM (Prolog based Tool for User Modelling) fue creada por Vergara H [Vergara, H., 1994]. En 1994, surge como un intento de crear un nuevo sistema, que mejore a los anteriores y a su vez, aúne las virtudes de sistemas anteriores como GUMS y UMT.

PROTUM es una herramienta escrita en el lenguaje de programación Prolog. PROTUM, de forma similar a como hacían sus antecesores, emplea estereotipos para el modelado de usuarios, sin embargo PROTUM es una herramienta mucho más potente que sus antecesores ya que si bien obtiene información de la jerarquía de estereotipos, no se limita a ella y a su vez, su sistema no se basa en tuplas “atributo, valor”.

2.2.4.TAGUS

TAGUS (Theory and Applications for General User/Learner-modeling Systems) fue desarrollado por Paiva y Self en 1994 [Paiva, A.& Self, J., 1995], esta herramienta al igual que sus antecesores, emplea estereotipos, para el modelado de usuarios. TAGUS fue desarrollado con dos objetivos principales, el primero consistió en desarrollar un marco en el que se representaran modelos de usuarios y de usuarios de aprendizaje, donde las actividades relacionadas con el conocimiento de estos, fuesen representativas. El segundo objetivo trata de incluir en el sistema mecanismos y técnicas para el modelado de usuario, como un servicio del sistema. Por tanto la idea principal de TAGUS, a diferencia de sus antecesores, fue llevar a cabo un banco de trabajo en el que varias técnicas y enfoques para el modelado de usuarios fuesen implementados y aplicados. De esta manera TAGUS consigue ofrecer a sus usuarios un

conjunto de servicios, que bien pueden ser empleados por usuarios, principalmente para testear los métodos de modelado y crear modelos, o bien pueden ser empleados por aplicaciones que usen los modelos de usuarios. Los servicios que TAGUS ofrece, bien para ser empleados por los usuarios o bien para ser empleados por aplicaciones, contienen mecanismos de mantenimiento de modelos de usuarios. Por tanto dentro del ciclo de funcionamiento de TAGUS, podemos identificar dos etapas principales: la de adquisición de conocimiento y la de mantenimiento del modelo. La arquitectura de TAGUS se compone de un modelo de aprendizaje de usuarios (ULM), un conjunto de funciones de mantenimiento de modelos, un motor de adquisición de conocimiento, un sistema de mantenimiento de los razonamientos y dos interfaces. TAGUS provee al mismo tiempo un lenguaje propio para definir los modelos, así como para testearlos, realizar simulaciones en el sistema. Este nuevo lenguaje, no solo se emplea para representar los modelos, sino que también es la forma de comunicación entre TAGUS y su entorno. Por tanto, de los modelos analizados hasta el momento, podemos concluir que TAGUS es seguramente el más completo, por lo menos en lo que a funcionalidad se refiere.

2.2.5.BGP-MS

BGP-MS (Belief, Goal and Plan Maintenance System) es un modelo desarrollado por Kobsa y Phol en 1995 [Kobsa, A. & Pohl, W., 1995], el modelo permite asumpciones a cerca del usuario y asumpciones de estereotipos a cerca de grupos de usuarios, dichas asumpciones, a diferencia de los modelos anteriores, pueden ser representadas en lógica de primer orden, lo cual supone una innovación con respecto a sistemas anteriores. Además de la ya mencionada anteriormente, este sistema introduce otras innovaciones que suponen una revolución en los modelos de usuario, entre dichas innovaciones podemos encontrar la posibilidad de ser empleado como un servidor de red, lo cual posibilita el funcionamiento como sistema multi-usuario y multi-aplicación, BGP-MS a diferencia de los sistemas anteriores, es también un sistema independiente de la aplicación o aplicaciones con las que ejecuta de forma concurrente, a su vez el sistema es capaz de adaptarse al dominio de trabajo, cosa que otros modelos de usuario no eran capaces de hacer. A continuación analizaremos cómo BGP-MS es capaz de llevar a cabo las nuevas funcionalidades que ofrece.

Para ajustar el comportamiento de BGP-MS a un dominio concreto, se deben seleccionar en la herramienta los componentes específicos para el dominio y alimentarlos con conocimiento relevante del dominio. A su vez, el funcionamiento de BGP-MS independiente de la aplicación, permite las siguientes posibilidades de funcionamiento del sistema, es decir entre la aplicación el sistema y el usuario: La aplicación puede informar a BGP-MS de opiniones o datos sobre el usuario, también podrá informar al sistema de las respuestas que el usuario da a

las preguntas formuladas durante la interacción, por último la aplicación puede pedirle a BGP-MS datos de inferencia sobre el conocimiento que se tiene del usuario.

El sistema está capacitado para responder a las peticiones de la aplicación, así como para formular a la aplicación nuevas preguntas necesarias para completar el modelo. Las preguntas que el sistema realiza permiten a su vez realizar inferencia o inferir nuevas reglas que aplicarán al usuario actual, en función del conocimiento que ya se tiene sobre éste.

2.2.6. DOPPELGANGER

El modelo de usuario Doppelganger, fue desarrollado por Jon Orwant en 1995, se encuadra al igual que el modelo anterior en los servidores de modelo de usuario que emplea un software de sensores. Por tanto el sistema almacena el conocimiento sobre el comportamiento del usuario mediante sensores, posteriormente crea un perfil para dicho usuario. Para la inferencia, el sistema emplea técnicas para generalizar y extrapolar los datos que recibe de los sensores, entre dichas técnicas podemos incluir, predicción lineal, modelos de Markov y clustering de estereotipos. Permite a los usuarios, inspeccionar y editar los modelos de usuario creados.

2.3. Modelos orientados a web

A finales de los años noventa, el valor de la web personalizada, comenzó a tomar auge en el área del comercio electrónico. La personalización web o segmentación web posibilita el marketing orientado a cada cliente, las promociones específicas para cada cliente y en general una segmentación antes no concebida, con todo esto empezó a producirse lo que se conoce como el comienzo del boom comercial y el marketing uno a uno. Se comprendió entonces el importante papel que los modelos de usuario jugaban dentro de este entorno creciente, a continuación se detallan algunas de las principales herramientas que surgieron dentro del campo de los modelos de usuario.

2.3.1. Group lens

Group Lens fue desarrollado por Net Perceptions en el año 1997 [Konstan, J. A., et. al., 1997], este sistema empleaba varios algoritmos de clustering colaborativo, para predecir en páginas de comercio electrónico. Las predicciones realizadas por el sistema se basan en un sistema de ratings explícitamente dados por los usuarios, otros ratings implícitos se derivan de datos de navegación de los usuarios, tales como productos que los usuarios revisan online o historial de transacciones.

2.3.2. LikeMinds

El sistema LikeMinds es parecido a Group Lens, fue desarrollado por Andromeda también en el año 2000. Las principales diferencias con el sistema Group Lens están en una arquitectura más modular, mejor distribución de carga, soporte para ODBC y los tipos de datos de entrada que son ligeramente diferentes a los de Group Lens, aunque también se basan en datos de navegación, datos sobre intereses de productos o preferencias dadas por los usuarios. Teniendo en cuenta que LikeMinds proporciona soporte para ODBC, podemos empezar a intuir el importante papel que a partir de este momento jugaran las bases de datos en el modelado de usuarios y su almacenamiento.

2.3.3. Personalization Server

Fue desarrollado por ATG en el año 2000, al igual que los dos modelos analizados previamente, permite la definición de reglas que asignan un usuario a uno o más grupos de usuarios, basándose en datos demográficos, tales como género o edad por mencionar alguno. Las reglas también pueden ser definidas con el fin de crear asunciones individuales sobre el usuario o sobre su comportamiento en la navegación web. Todos estos datos son posteriormente empleados para personalizar el contenido web de ciertos sitios, en función de los usuarios visitantes. De esta manera, se entiende que la personalización realizada por este sistema se acerca mucho al enfoque de modelos de usuarios clásicos.

2.3.4. FrontMind

Sistema creado por Manna en el año 2000, provee un desarrollo basado en reglas, gestión, simulación y personalización del entorno basada en información personalizada y servicios web personalizados. El sistema FrontMind tiene valores distintivos con respecto a otros sistemas similares, como Personalization Server, ya que cuenta con redes Bayesianas para modelar el comportamiento de los usuarios en su marco de personalización.

2.3.5. Learn Sesame

Learn Sesame fue desarrollado conjuntamente por Open Sesame y Bowne en el 2000, permite definiciones del dominio del modelo basándose en objetos, atributos de dichos objetos y eventos. Acepta información acerca del usuario proveniente de la aplicación, categoriza dicha información según el modelado del dominio y trata de descubrir patrones recurrentes, correlaciones y similitudes a lo largo del un clustering incremental. Lo que el sistema considera observaciones de interés serán devueltas a la aplicación.

2.4. Modelos probabilísticos

A continuación vamos a exponer los principales modelos probabilísticos, así como los distintos enfoques dentro de los modelos de usuario probabilísticos. Los modelos probabilísticos, se basan en probabilidades o certezas de cada dato que se almacena de los usuarios, son por esto radicalmente distintos a los modelos vistos hasta ahora.

Dentro de los modelos probabilísticos y de la investigación llevada a cabo en esta área, se pueden diferenciar dos enfoques adoptados, los modelos probabilísticos basados en contenidos y los modelos probabilísticos colaborativos, a continuación se expone un breve análisis de ambos enfoques antes de comenzar con los modelos propiamente dichos.

2.4.1. Modelos probabilísticos basados en contenidos y modelos probabilísticos colaborativos

Los modelos probabilísticos basados en contenidos, son empleados cuando el comportamiento pasado de un usuario puede ser empleado como un indicador fiable del comportamiento del usuario en el futuro. En este enfoque los modelos se construyen a partir de los datos del comportamiento pasado del usuario. Por ejemplo, podríamos considerar el dominio de recomendaciones de películas y suponer que existe un usuario (Fred) quien indica que le gusta “Star wars”, “Raiders of the Lost Ark” y “Air Force One”, el modelo basado en contenido aprende los tipos de películas que le gustan al usuario (Fred), véase películas de acción con Harrison Ford, en función de estos datos y este conocimiento aprendido el sistema podría recomendar otras películas similares como “Witness”. Los modelos basados en contenidos son especialmente adecuados para entornos en los que los usuarios tienden a mostrar comportamientos idiosincrásicos, es decir, con rasgos distintivos propios de un individuo o una comunidad. Sin embargo, los modelos basados en contenidos requieren de sistemas capaces de almacenar grandes cantidades de datos para posibilitar la construcción del modelo.

Los modelos probabilísticos colaborativos, son empleados cuando no se puede asumir que un usuario se comportara de la misma forma que otros usuarios. En este enfoque el modelo se construye a partir de los datos de un grupo de usuarios y a partir de este modelo se realizan predicciones sobre un usuario individual. Volviendo al ejemplo anteriormente expuesto de las recomendaciones de películas y el usuario (Fred), el sistema debe ser capaz de inferir que el gusto de Fred es similar al de un determinado grupo y según esto, recomendara al usuario otras películas que los usuarios de ese grupo valoran positivamente. El enfoque de modelos colaborativos es muy útil para realizar predicciones sobre un usuario cuando este fue

identificado como miembro de un grupo concreto, o bien para predicciones sobre un usuario conocido en una nueva situación. Por ejemplo, imaginemos el caso de un usuario visitando una página web. En el caso de desconocimiento, el modelo colaborativo, emplearía la información del resto de usuarios que visitaron la misma página, para predecir la pagina que es más probable que el usuario requiera a continuación.

Una vez analizados los dos principales enfoques de modelos probabilísticos, pasamos a ver los principales modelos que siguieron ambos enfoques. Se debe tener en cuenta que muchos de los modelos que se van a exponer fueron usados bajo los dos enfoques anteriormente expuestos, los podemos clasificar de la siguiente forma:

- Modelos lineales
- Modelos TFIDF
- Modelos de Markov
- Modelos de redes neuronales
- Modelos de clasificación
- Modelos de redes Bayesianas

2.5. Modelos lineales

Los modelos lineales siguen una estructura sencilla, lo cual facilita su extensión, generalización y aprendizaje. Los modelos lineales se basan en sumas de pesos para valores conocidos para producir un peso para un valor desconocido. Los modelos lineales fueron empleados siguiendo los dos enfoques expuestos anteriormente. Estos modelos fueron desarrollados en su mayoría a mediados de los años noventa principalmente por Jon Orwant.

2.6. Modelos TFIDF

Los modelos TFIDF (Term frequency inverse document frequency) emplean un método de pesos comúnmente usado en recuperación de información para encontrar documentos que cumplen con una determinada query. Este método representa un documento como un vector de pesos, donde cada peso corresponde con un término del documento. La similitud entre dos documentos o un documento y la query, se mide con el coseno del ángulo formado por los correspondientes vectores, de ambos documentos.

Balabanovic en 1998 y Moukas y Maes en el mismo año aplicaron modelos TFIDF en modelos basados en contenidos, los cuales recomendaban a los usuarios documentos en función de otros documentos de interés para dicho usuario.

Moukas y Maes extendieron estos enfoques empleando algoritmos genéticos para adaptar de forma automática sus sistemas de recomendación.

2.7. Modelos de Markov

De la misma forma que ocurre con los modelos lineales, los modelos de Markov [DRH Miller, T Leek & RM Schwartz, 1999], cuentan con una estructura muy sencilla. El motivo por el que la estructura de estos modelos es tan sencilla, es por su fiabilidad en la asumpción de Markov, la cual representa secuencias de eventos y asume que la ocurrencia del siguiente evento depende sólo de un número fijo de eventos anteriores. Dado un número de eventos observados, el siguiente evento se predice en función de la distribución de probabilidad de los eventos que en el pasado han seguido a los observados.

Por ejemplo, cuando la tarea consiste en predecir la siguiente página web que solicitará un usuario, el último evento observado puede consistir únicamente en la última página web visitada o podría contener información adicional, como el enlace que llevo al usuario a esa página web.

Bestavros en 1996, Horvitz en 1998 y Zukerman en 1999, emplearon modelos de Markov bajo el enfoque de los modelos colaborativos, para predecir peticiones de sitios web. El modelo de Bestavros, calcula la probabilidad de que un usuario solicite una determinada página web en el futuro.

El modelo de Horvitz calcula la probabilidad de que un usuario solicite un determinado documento en el futuro. Por último el modelo de Zukerman compara el funcionamiento de varios modelos de Markov, para predecir las páginas web requeridas por un usuario.

Las predicciones generadas por estos modelos, fueron posteriormente empleadas para modelos de anticipación, que eran capaces de enviar a los usuarios los documentos o sitios web de mayor relevancia.

2.8. Modelos de redes neuronales

En el caso de los modelos de redes neuronales, se emplea esta técnica tomada de la Inteligencia Artificial. Se trata de modelos dinámicos y auto configurables, lo cual facilita mucho el mantenimiento de estos modelos. Las redes neuronales, son capaces de expresar una gran variedad de superficies de decisión no lineales, esto lo hacen gracias a la estructura de la red, gracias a umbrales no lineales y gracias a diferentes pesos que se asignan a las ramas entre los nodos de la red.

En 1993 Jennings, A. y Higuchi, H. emplearon los modelos de redes neuronales para representar las preferencias de los usuarios en relación a un conjunto de artículos.

Por cada artículo, existía una red neuronal, donde cada nodo de la red representa términos que aparecen en distintos artículos marcados como relevantes por el usuario. Las ramas entre los nodos representan la fuerza de la asociación entre términos que aparecen en un mismo artículo.

2.9. Modelos de clasificación

Los métodos de clasificación, particionan un conjunto de objetos en clases, en función de los valores de los atributos de dichos objetos. Dado un espacio n-dimensional que corresponde con el conjunto de atributos relevantes, el cluster de clases generadas contiene ítems que están cerca los unos de los otros dentro de este espacio y lejos de otros clusters. Por lo tanto lo parecidos que son dos objetos es inversamente proporcional a la distancia entre ellos.

Los métodos de clasificación son no supervisados, por lo tanto no existe información a priori de la clase a la que debe pertenecer un objeto en concreto.

Perkowitz, M. y Etzioni, O. en el año 2000 y siguiendo el enfoque de modelos colaborativos, emplearon una modificación del clustering tradicional para automáticamente crear índices de páginas que contienen enlaces a otras páginas, es decir, paginas relacionadas entre sí. Se entiende que estas páginas son buenas candidatas para ser visitadas por un usuario dentro de una misma sesión. Esta técnica de modificación de clustering que adoptó el nombre de cluster mining, permitía documentos solapados en varios clusters.

2.10. Modelos de redes Bayesianas

Las redes bayesianas y varias extensiones de estas, han crecido rápidamente dentro del campo de la Inteligencia Artificial y han sido usadas para varias tareas en el modelado de usuarios. Las redes bayesianas consisten en grafos dirigidos acíclicos donde los nodos corresponden a un conjunto de variables. Los nodos están conectados por arcos que pueden ser interpretados como enlaces de nodos padres a nodos hijos. Cada nodo tiene una distribución de probabilidad asociada que corresponde a las combinaciones de valores de sus nodos padres.

Los modelos basados en redes bayesianas, son más flexibles que los mencionados hasta el momento ya que permiten una representación compacta de cualquier distribución de probabilidades y permiten realizar predicciones sobre un conjunto de variables, más que sobre

una sola variable que es el comportamiento más adoptado por los modelos probabilísticos analizados hasta el momento.

Una propiedad importante de las redes bayesianas es que soportan sin demasiados cambios el enfoque de modelos colaborativos, así como el enfoque de modelos basados en contenido. El enfoque colaborativo normalmente se emplea para obtener tablas de probabilidad condicionales y las primeras creencias de la red bayesiana. Estas creencias iniciales pueden ser actualizadas mediante un enfoque basado en contenidos, una vez que el usuario está usando la red.

El modo de operación expuesto anteriormente posibilita al modelo superar los problemas de recolección de grandes cantidades de datos del enfoque basado en contenidos, mientras que al mismo tiempo se adapta a un aprendizaje colaborativo.

Por tanto podemos decir que a lo largo de los años las redes Bayesianas y sus extensiones, han sido empleadas con éxito en diversas tareas de predicción. Por ejemplo en 1999 Lau y Horvitz construyen modelos basados en Redes Bayesianas para predecir la próxima query que un usuario de Internet formularía. En el año 1998 Albrecht comparó el funcionamiento de varias redes Bayesianas dinámicas que eran capaces de predecir la próxima acción de un usuario, su próxima localización y la misión actual, en un juego de aventuras multiusuario. Por mencionar otro ejemplo de uso de redes Bayesianas para tareas de predicción, en el año 2000 Jameson las uso con el fin de predecir la tasa de error de un conjunto de usuarios, que debían cumplir unas instrucciones previamente dadas.

A parte de los ya mencionados, existen otros muchos casos de aplicación de redes Bayesianas, que a lo largo del tiempo se muestran como un sistema exitoso en tareas de predicción y por lo tanto en la tarea de modelado de usuarios.

2.11. Otros modelos de usuario

La siguiente sección, describe el modelo de usuario llevado a cabo en el proyecto mercurio, el cual no se encuadra en ninguna de las categorías anteriores.

2.11.1. El proyecto Mercurio: servidor personalizado de información

EL proyecto Mercurio trata de construir un servidor personalizado de noticias con una representación del usuario más elaborada que las empleadas hasta el momento. Inicialmente es el usuario el que indica al sistema sus preferencias a la hora de recibir noticias, de manera que con dichas preferencias se genera un modelo de usuario que se utilizara para enviarle las

noticias más relevantes al usuario. El servidor creado, coopera también con un buscador que permite a los clientes acceder de forma inteligente a las noticias del día mediante búsquedas complejas a través de la web.

Las interacciones del usuario con el sistema son todas analizadas y se extrae información de todas ellas, dicha información es empleada para realimentar el sistema.

A continuación y puesto que este modelo emplea una representación del usuario más elaborada que los anteriores, vamos a analizarla con un poco más de detalle.

- Estereotipos: En el caso base, se representa al usuario a base de clasificarlo. Los datos iniciales que facilita el usuario, lo clasifican en un estereotipo.
- Términos representativos: Se le asigna al usuario un conjunto de palabras clave que, o bien aparecen habitualmente en sus consultas, o bien aparecen con relativa frecuencia en los documentos que el usuario consulta. Este método se hace de forma automática manteniendo un histórico de las interacciones del usuario. La desventaja de esta técnica es que puede inducir a errores por la polisemia de ciertas palabras. Para solventar este error pueden emplearse bases de datos léxicas para desambiguar.
- Jerarquía de categorías del sistema: El sistema es capaz de asignar categorías a cada noticia. El usuario posteriormente puede indicar si una categoría es de su interés o no, esta información se almacena en su modelo de usuario y se emplea en la toma de decisiones. A cada una de estas asignaciones de interés por una categoría se le da una certidumbre, que indica en futuras decisiones el peso de esa información, dicha certidumbre se irá modificando en función de las interacciones del usuario. Por ejemplo, si un usuario marca como interesante una categoría pero posteriormente rechaza varias noticias de esta categoría, la certidumbre de esta disminuirá.
- Palabras clave: El usuario puede indicar palabras clave que o bien son de su interés, o bien no lo son y estas también pesarán según una certidumbre en la toma de decisiones.
- Eliminación de información poco utilizada: Para evitar el crecimiento de los modelos de usuario, y de la información de estos, el sistema elimina de cada modelo aquellos términos, que no aparecen en las últimas búsquedas del usuario.

3. Objetivos de la propuesta

La propuesta que se detalla en el presente documento, comprende el desarrollo de un modelo de usuarios de grupo, el modelo propuesto se enmarca dentro de los modelos de grupos genéricos, es decir no se trata de un modelo enfocado a un dominio concreto, sino que se trata de un modelo basado en la experiencia. El modelo propuesto por tanto permite a un sistema, adaptarse a las características de cada usuario particular, el modelo es independiente de la aplicación que lo emplee y provee información suficiente para que el sistema sea "customizable", es decir adaptable a las necesidades de los usuarios.

El modelo propuesto, por lo tanto trata de alejarse de la mayoría de las propuestas analizadas en el capitulo anterior del presente documento, que o bien se enfocaban en modelos de perfiles, o bien se enfocaban en modelos adaptados a un dominio concreto. Puede afirmarse por tanto que el proyecto de investigación que aquí se plantea, pretende alejarse de la mayoría de herramientas existentes hasta el momento y analizara distintos mecanismos y arquitecturas, con el fin de alcanzar una solución de compromiso que nos permita mejorar los sistemas similares existentes.

A continuación, vamos a describir brevemente las dos alternativas planteadas para la construcción del modelo de grupos, así como los mecanismos básicos que emplean dichas propuestas. En el presente proyecto, se van a validar varios modelos de usuario de grupos genéricos, ambos con distintos mecanismos y distinto planteamiento, con el fin de analizar cuál de los dos planteamientos es más adecuado. Por tanto las propuestas planteadas, difieren tanto en los mecanismos, como en el enfoque inicial de cada una de ellas.

Pasamos por tanto a definir el planteamiento de ambas propuestas para posteriormente analizar a muy alto nivel los diferentes mecanismos que emplea cada propuesta. En cuanto al planteamiento de las propuestas, destacar que existen varias formas de tratar e interpretar la información y la noción de cuan parecidos son dos grupos de usuarios, el proyecto que nos ocupa analizara varias propuestas de las evaluara con el fin de decidir cuál de ellas es la mejor.

En lo que se refiere a los mecanismos básicos de ambas propuestas, estos pueden resumirse en dos:

- Mecanismo de consulta
- Mecanismo de aprendizaje

Cada uno de los anteriores mecanismos, está basado en formulas matemáticas, que permitirán evaluar el correcto funcionamiento de cada uno de los mecanismos, así como del sistema en global.

A su vez cada uno de estos mecanismos básicos que se plantea construir, se descompone en varios submecanismos más específicos, en los que se apoya para llevar a cabo su cometido.

3.1. Mecanismo de consulta

Requiere de los siguientes mecanismos para poder realizar las tareas de consulta del sistema, mecanismo de encaje y mecanismo de inferencia.

3.2. Mecanismo de aprendizaje

Requiere al igual que el mecanismo de consulta de un mecanismo de encaje y a su vez de un mecanismo de fusión.

Por lo tanto, tal y como se explico, ambas propuestas cuentan con dos mecanismos básicos, consulta y aprendizaje, de igual forma las dos propuestas cuentan con un conjunto de submecanismos, que son empleados por los mecanismos básicos, dichos submecanismos son, encaje, inferencia y fusión, a continuación daremos una noción somera del cometido que realizara cada uno de los submecanismos en ambas propuestas y para cada mecanismo básico que hace uso de ellos.

3.2.1. Submecanismo de encaje

Si bien el nombre de mecanismo de encaje, es el que suele darse a esta herramienta en el ámbito del modelado de usuario, su cometido es el de dar una noción de cómo de parecidos son dos usuarios o grupos de usuarios, este submecanismo es empleado tanto por el mecanismo de consulta, como por el de aprendizaje.

3.2.2. Submecanismo de inferencia

Lo emplea el mecanismo de consulta, para determinar nueva información relevante para un usuario, esta información no viene dada de forma explícita por el usuario, sino que se obtiene por similitud entre un usuario y un grupo de usuarios.

3.2.3. Submecanismo de fusión

Encargado de fusionar en un solo grupo, dos grupos de usuarios, es empleado por el mecanismo de aprendizaje, para fusionar información, en caso de crecer mucho el volumen de información del sistema.

Así pues, puede afirmarse que los mecanismo básicos planteados anteriormente, son los mismos para ambas propuestas, sin embargo los submecanismo en los que se apoyan ambos mecanismos, si bien son los mismos, emplean formulaciones matemáticas diferentes y diferente tratamiento de la información, lo que permite ajustar cada una de las propuestas a un funcionamiento concreto y evaluarlas con el fin de determinar cuál de ellas se comporta mejor.

Tras el análisis de objetivos realizado en el presente apartado, se comienza con el análisis del proyecto, en el que se introducirán las nociones necesarias para conocer el problema que se plantea más en detalle, así como las alternativas y posibilidades analizadas.

4. Metodología

4.1. Metodología del desarrollo

La presente sección del documento describe la metodología de desarrollo seguida durante el proceso de desarrollo software. El proceso de desarrollo software, para el caso que nos comprende, siguió una metodología de ciclo de vida en espiral, esta metodología plantea varias iteraciones, donde cada iteración representa un conjunto de actividades, las actividades o tareas no están prefijadas, sino que se escogen en función de las realizadas en iteraciones anteriores y de los riesgos de estas.

Tras analizar brevemente de forma genérica, la metodología empleada, se analizarán los detalles de cómo se aplico la metodología citada al caso concreto del proyecto. Para ver con mayor claridad cómo se aplico la metodología, se adjunta la siguiente imagen del ciclo de vida en espiral.

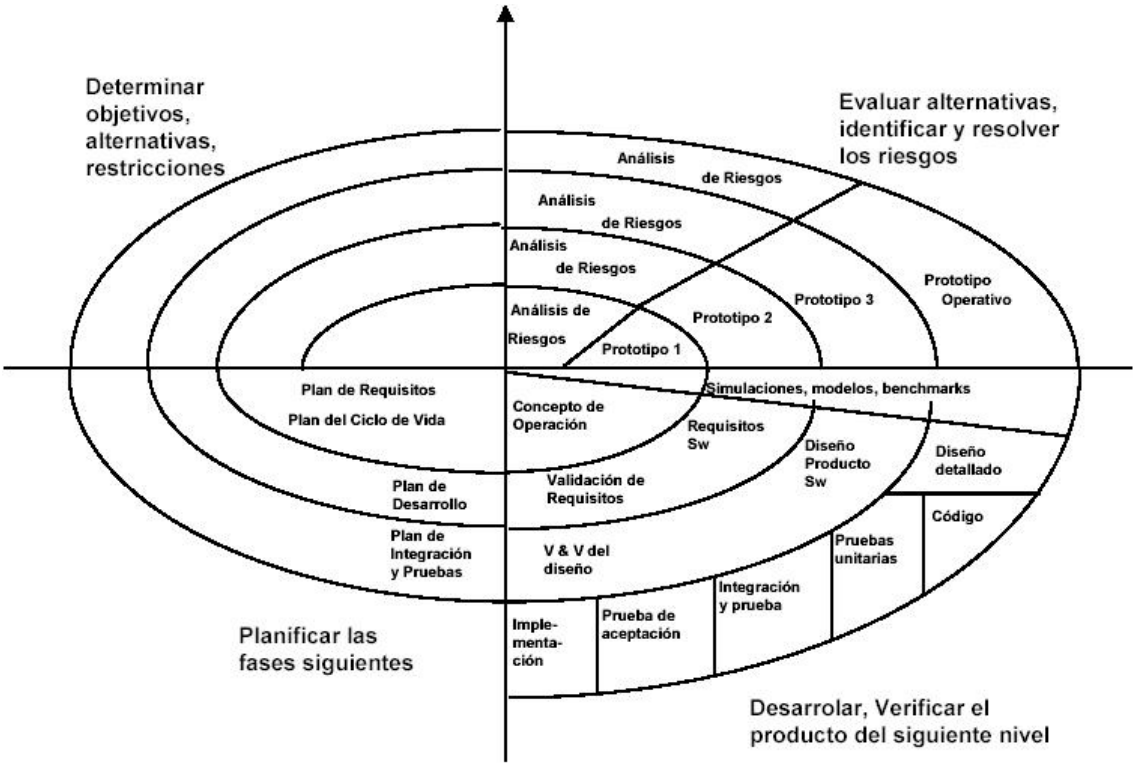


Ilustración 01. Ciclo de vida en espiral

Como puede observarse en la figura 01 y tal como se menciono con anterioridad, este tipo de ciclo de vida se desarrolla en varias fases, con tareas no prefijadas, para el caso del proyecto que nos ocupa: El desarrollo se dividió en cuatro grandes fases, a continuación se detallan las tareas que se realizan en cada fase. En esta primera aproximación, se plantea un análisis del ciclo de vida a priori, para posteriormente en el apartado conclusiones de esta sección, introducir un análisis a posteriori del ciclo de vida.

4.1.1. Primera fase o creación del primer prototipo

Durante esta primera fase, se pretende llevar a cabo varias tareas como pueden ser, un análisis de objetivos, un planteamiento del problema, una tarea de análisis, donde se plantean las distintas alternativas, una fase de diseño, una fase de implementación y por último una validación del prototipo con datos ficticios.

Así pues a lo largo de esta primera fase del ciclo, se trata de analizar el problema, llegar a una posible solución al problema, que cuente con un tratamiento de información valido y una formulación matemática válida para los mecanismos básicos planteados en el apartado anterior de esta memoria. Por tanto esta en esta primera fase se pretende desarrollar un prototipo básico, que no estará enfocado a la eficiencia, sino a la eficacia. Se trata por tanto de conseguir

formulaciones matemáticas adecuadas para cada uno de los procedimientos y sub procedimientos detallados anteriormente. Dichas formulaciones matemáticas, se enfocarán a proveer valores dentro de los rangos esperados en cada mecanismo. Por lo tanto en los objetivos de esta primera fase del ciclo de vida, no se contempla la mejora de los mecanismos del modelo. Finalmente en esta primera fase se debe validar el prototipo creado con datos ficticios introducidos a mano por el desarrollador. Por lo tanto, esta primera validación del modelo se llevara a cabo con un conjunto muy reducido de datos que permitirán validar el correcto funcionamiento del prototipo, así como de los mecanismos de este.

4.1.2. Segunda fase o alternativas propuestas

Durante la segunda fase o segundo ciclo, se propone analizar en detalle los resultados obtenidos en la fase anterior y plantear alternativas que al menos de forma teórica puedan mejorar los resultados obtenidos en la fase anterior. Por lo tanto, en esta segunda fase del ciclo de vida, partiendo de un prototipo validado en lo que a su estructura y funcionamiento se refiere, se trata de mejorar los planteamientos matemáticos de todos los mecanismos del modelo, con el fin de mejorar el funcionamiento y tratamiento de la información del prototipo.

Así pues en esta segunda fase, se llevarán a cabo planteamientos matemáticos que al menos de forma teórica permitan demostrar que el prototipo realiza un mejor tratamiento de los datos. De la misma forma que sucedía con el primer prototipo, este segundo será validado con datos ficticios, concretamente con los mismos datos con los que se validó el primer prototipo, con el fin de poder valorar las posibles mejoras del prototipo.

4.1.3. Tercera fase o datos reales

En la tercera fase del ciclo de vida seguido, se pretenden validar los modelos creados empleando datos reales en lugar de datos ficticios. Por lo tanto esta etapa, ofrecerá resultados cuantitativos de cómo se comporta el modelo al emplear datos reales. Los datos con los que se validará, fueron extraídos mediante un web crawler de una red social. Los resultados obtenidos en esta etapa, serán por tanto mucho más fiables que los obtenidos en etapas anteriores, por tratarse de experimentación con datos reales. Para finalizar la tercera fase del ciclo de vida, se analizarán los resultados obtenidos tras la experimentación.

4.1.4. Cuarta fase o prototipo final

Durante la cuarta fase, se pretende crear dos prototipos con distinto planteamiento teórico y por tanto con distinto tratamiento de los datos y validar ambos modelos con los datos reales con los que se experimento en la fase anterior. Finalmente los resultados de ambas

experimentaciones serán analizados con el fin de determinar cuál de los planteamientos llevados a cabo realiza mejor tratamiento de la información y reporta por tanto mejores resultados.

4.1.5. Conclusiones

En este apartado se pretende analizar de nuevo las fases del ciclo de vida del software, tratando de explicar a posteriori cuales fueron las tareas llevadas a cabo durante el desarrollo software. Por tanto en la presente sección, se ofrece una visión más real de cuál fue el desarrollo software y cuáles fueron las tareas llevadas a cabo en cada fase del ciclo software.

Así pues, tal y como se explico anteriormente, en la primera fase, se desarrollo un primer prototipo, que manejaba valores de certidumbre en el rango [0,1], el prototipo fue validado con datos ficticios. Tras analizar los resultados de la primera fase, se planteó un nuevo prototipo en la segunda fase, el segundo prototipo, el cual maneja datos de certidumbre en el rango [-1,1], es decir, a diferencia del primer prototipo que discriminaba valores de certidumbre parecidos, este segundo prototipo es capaz de ofrecer una noción, no solo de cuan parecidos son dos valores de certidumbre, sino que también aporta información de valores de certidumbre muy dispares. Al mismo tiempo, las formulaciones matemáticas de los mecanismos básicos del prototipo fueron ajustadas al nuevo rango matemático, con lo que se obtuvo un nuevo prototipo que a priori debe funcionar mejor que el primero desarrollado. En esta segunda fase, tras la implementación del prototipo, este también se valido con datos ficticios.

En la tercera fase tal y como se planteo anteriormente, se validaron ambos prototipos con datos reales; tras una experimentación que duró unos tres meses, se evaluaron los resultados que no fueron del todo satisfactorios, por esto se propuso mejorar ambos prototipos en la cuarta y definitiva fase del proyecto. Tras analizar los problemas encontrados en esta tercera fase, se revisaron los planteamientos matemáticos empleados y se propusieron mejoras a todos los mecanismos del sistema.

En la cuarta fase del desarrollo software, se realizo un nuevo planteamiento matemático para los dos prototipos creados, dicho planteamiento se enfocó a solucionar los problemas encontrados en la tercera fase. Entre los problemas encontrados, podemos destacar el reparto de usuarios en grupos durante el aprendizaje del sistema. Este reparto provocaba la creación de grupos de usuarios muy grandes y grupos muy pequeños, o de un solo usuario. Para solucionar este inconveniente, se planteo una nueva formulación matemática en la que se reajustaron pesos y expresiones matemáticas. Tras la implementación de los prototipos finales, se reanudaron las experimentaciones con datos reales y se mejoraron los resultados obtenidos en experimentos anteriores.

Así pues las dos versiones de prototipos obtenidas y validadas en esta cuarta fase del proyecto, se consideran versiones finales, que superaron la validación con datos reales.

5. Análisis y diseño

La presente sección, describe el proceso de análisis que se llevó a cabo, con el fin de plantear las distintas alternativas al problema que nos ocupa. En esta sección, se plantearán alternativas al problema, se realizará un estudio del problema y del entorno de este. A su vez, por tratarse de una sección conjunta de análisis y diseño, la sección actual, decidirá cuál es la mejor alternativa, para cada uno de los problemas planteados y justificará las decisiones de diseño tomadas.

Tal y como se detalló con anterioridad, el ciclo de vida seguido a lo largo del proyecto es en espiral, este ciclo de vida permite en cada fase ir comprobando las anteriores, realizar un análisis de riesgos de las etapas anteriores y planificar las próximas fases, en el caso del análisis del problema, se seguirá el mismo proceder, siendo por tanto coherentes con el ciclo de vida seguido.

La etapa de análisis del proyecto, resulta de gran importancia para el éxito del proyecto, ya que los resultados de dicha fase, van a incidir en etapas posteriores, como por ejemplo en el diseño del sistema. Por tanto un mal análisis del problema, desembocaría en un mal diseño y probablemente en el fracaso del proyecto, por consiguiente podemos afirmar que nos encontramos en una de las etapas más importantes del proyecto.

Así pues los resultados obtenidos durante la fase de análisis, tendrán repercusión directa en la siguiente fase del proyecto o etapa de diseño. Si bien durante esta fase se plantearán alternativas, todas ellas orientadas a alcanzar la mejor solución, para el problema concreto que nos ocupa.

Dentro de la fase de análisis, vamos a descomponer el problema en varios subproblemas, el primero de ellos abarca la tarea de crear el modelo de usuario en sí mismo, el segundo subproblema, consiste en crear un cliente que sea capaz de gestionar los experimentos sobre el modelo de usuario creado y a partir de una base de datos con los datos de origen.

Tanto el análisis como el diseño del problema, se dividen en tres grandes puntos: análisis y diseño funcional, de necesidades de información y físico. En la sección funcional se detallarán los aspectos funcionales del sistema, las alternativas de funcionalidades y la solución escogida. En cuanto al aspecto de necesidades de información, se analizará la información que debe ser capaz de almacenar el sistema, por último, el aspecto físico ofrece nociones de la distribución física del proyecto, así como de sus comunicaciones.

5.1. Análisis y diseño funcional

En esta sección, se pretenden abstraer todos los aspectos de implementación y tecnología, con el fin de ofrecer en detalle qué es lo que debe hacer el sistema, cuáles son las posibles alternativas para las distintas funcionalidades del sistema y explicar la alternativa escogida, justificando esta decisión.

En primera instancia se va a descomponer el módulo del modelo de usuario, en los subsistemas que lo componen, con el fin de poder analizar posteriormente cada uno de los módulos por separado. La siguiente figura descompone en módulos el sistema.

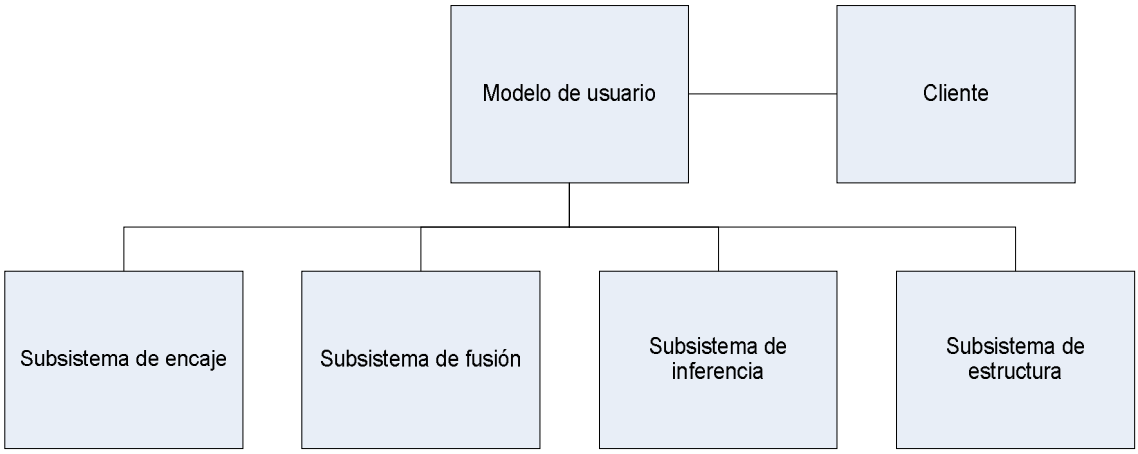


Ilustración 02. Subsistemas modelo de usuario

La figura anterior (ilustración 02), descompone el sistema de modelo de usuario, en varios subsistemas más sencillos, de forma que la tarea de análisis sea más fácil y comprensible. A continuación se procede a describir brevemente cada uno de los subsistemas expuestos anteriormente.

5.1.1. Subsistema de encaje

El subsistema de encaje, es el encargado de dados dos grupos de usuarios, proveer información de cómo de parecidos son dichos grupos de usuarios, es decir este subsistema, permite dado un usuario actual, determinar el grupo de usuarios del modelo más parecido a dicho usuario. El subsistema de encaje por lo tanto, debe ser capaz de llevar a cabo las siguientes tareas:

- Almacenamiento de información: el subsistema debe ser capaz de almacenar toda la información de los dos grupos de usuarios sobre los que se realizará la tarea de encaje. Por tanto, para cada tarea de encaje el subsistema debe mantener dos estructuras en las que almacenará la información de los dos grupos de usuarios sobre los que se desea calcular el encaje. Para el cálculo del encaje, se plantean dos alternativas en lo que al almacenamiento de información se refiere.

  - Almacenar la información en memoria: mantiene los usuarios en memoria mientras se calcula el encaje.
  - Almacenar la información en base de datos: Recupera los usuarios de la base de datos para realizar los encajes.

Finalmente se decidió por cuestiones de eficiencia mantener los usuarios en memoria ya que los múltiples accesos que el módulo de encaje debe realizar para recuperar la información de los usuarios, son más rápidos si se realizan sobre memoria que si se realizan sobre la base de datos del modelo.

- Categorizar la información: entre las funciones de este subsistema se incluye la categorización de la información anteriormente almacenada. Así pues, este subsistema debe clasificar la información almacenada de los usuarios. La información de los usuarios se divide en tuplas formadas por un identificador, una característica, un valor y una certeza. De esta forma, cada característica se debe clasificar en uno de los siguientes casos disjuntas, coincidentes y equivalentes. A continuación exponemos la definición de encaje con el fin de comprender la división de características expuesta.
- Sean los identificadores de grupos de usuario, Id_i, Id_j, para ambos identificadores tenemos un conjunto de características, $C_{ij} = \{C_i, C_j\}$, donde cada C_i toma valores en V_i, siendo Vi los valores de las características de C_i, y siendo V_j el conjunto de valores de C_j. Asi pues la formulación matemática anterior queda de la siguiente forma: $C_{ij} = \{C_i \in Vc_i, C_j \in Vc_j\}$. A su vez tenemos la cardinalidad del conjunto anterior que identificamos como $card(C_{ij})$. Y por último la función $Cert(C_{ij})$ que nos devuelve la certeza de dicha característica.
- Según esto, clasificamos las características de cada grupo de usuarios de la siguiente forma:
  - *Características disjuntas*: Una característica es disjunta si se cumple lo siguiente. Para toda $C_k \in C_{ij}$: Si V_{Ck} no se encuentra en V_{ci} y V_{Ck} no se encuentra en V_{cj} entonces la característica es disjunta.

- *Características coincidentes*: Si para toda C_k € C_{ij}: Si V_{Ck} € V_{Ci} y V_{Ck} € V_{cj} y $V_{ci} \neq V_{cj}$ (existe al menos un valor distinto) entonces las características son coincidentes. Es decir coinciden las características pero no los valores.
- *Características equivalentes*: Dos características son equivalentes si para toda C_k € C_{ij}:

  Si C_k € V_{ci} y C_k € V_{cj} y existe v1 tal que (C_k, v1) € V_{ci} y (C_k, V1) € V_{cj} y V_{ij} = {V_i tal que (C_k, V_i, Z_i) € V_i o V_j tal que (C_k, V_j, Z_j) € V_j}

  Según lo expuesto anteriormente, el subsistema de encaje, debe ser capaz de clasificar cada característica de cada grupo de usuarios, en uno de los casos anteriores.

- Cálculo del encaje: en función de la clasificación de características anteriormente dada y empleando los mecanismos de encaje, basados en expresiones matemáticas, el subsistema de encaje debe ser capaz de calcular un valor de encaje o parecido entre los dos grupos de usuarios. Tal y como se expuso en la sección de metodología, se trabaja con dos modelos paralelamente, cada uno de ellos realizará un tratamiento de valores distinto, por lo que tendremos un modelo que manejará datos en el rango [0,1] y otro que manejará datos en el rango [-1,1] para los valores de las certezas calculados en los distintos procesos del modelo de usuarios. A continuación se exponen las expresiones matemáticas empleadas para cada uno de los tipos de características vistos anteriormente.

  - Características disjuntas:
    - Modelos en [0,1] y [-1,1]
      - Cert(C_k) = 0
  - Características coincidentes:
    - Modelo en [0,1]
      - Cert(Ck) = (1-2*Zk) / ((card(Zk)-1) siendo Zk la certeza de la característica Ck
    - Modelo en [-1,1]
      - Cert(Ck) = -Zk / ((card(Zk)-1) siendo Zk la certeza de la característica Ck

- Características equivalentes: Para el caso de características equivalentes se diferencian varios casos en función de los valores de las certezas de cada característica, así pues podemos tener, equivalente nulo, equivalente de distinto signo, equivalente de igual signo positivo y equivalente de igual signo negativo, a continuación se expone que deben cumplir las distintas certezas, para encuadrarse en uno u otro caso.

  - Modelo en [0,1]

    - Equivalente nulo: $Z_k * Z_i = 0$
      - $Cert(Z_k) = 0$
    - Equivalente de distinto signo: $Z_k * Z_i < 0$
      - $Cert(Z_k) = -| Z_k - Z_i|$
    - Equivalente de igual signo positivo: $Z_k * Z_i > 0$
      - $Cert(Z_k) = | Z_k + Z_i -1|$
    - Equivalente de igual signo positivo: $Z_k * Z_i > 0$
      - $Cert(Z_k) = | Z_k + Z_i -1|$
    - Equivalente de igual signo negativo: $Z_k * Z_i > 0$ AND $Z_k < 0$
      - $Cert(Z_k) = | Z_k + Z_i -1| / ((card(Z_k)-1)$ siendo Z_k la certeza de la característica C_k

  - Modelo en [-1,1]

    - Equivalente nulo: $Z_k * Z_i = 0$
      - $Cert(Z_k) = 0$
    - Equivalente de distinto signo: $Z_k * Z_i < 0$
      - $Cert(Z_k) = -| Z_k - Z_i| / 2$
    - Equivalente de igual signo positivo: $Z_k * Z_i > 0$
      - $Cert(Z_k) = | Z_k + Z_i| / 2$

- Equivalente de igual signo positivo: $Z_k * Z_i > 0$
  - $Cert(Z_k) = | Z_k + Z_i | / 2$
- Equivalente de igual signo negativo: $Z_k * Z_i > 0$ AND $Z_k < 0$
  - $Cert(Z_k) = (| Z_k + Z_i | / 2) / 2 * ((card(Z_k)-1)$ siendo Z_k la certeza de la característica C_k

- Retorno del encaje: la última función del subsistema, consiste en devolver el valor calculado de encaje entre los dos grupos de usuarios.

5.1.2. Subsistema de fusión

El subsistema de fusión, se encarga de realizar la fusión de dos grupos de usuarios en uno solo, esto se produce por un exceso de información en el modelo, que conlleva la necesidad de fusionar dos grupos en uno. Así pues, en caso de que se supere un umbral de información en el modelo, el usuario actual se fusionará con el grupo de usuarios al que más se parezca. Este módulo por tanto debe realizar las siguientes tareas:

- Seleccionar el grupo más parecido: Para llevar a cabo esta tarea, se identificaron varias alternativas, antes de analizarlas, vamos a exponer cual es el objetivo que se pretende. Una vez es necesario fusionar dos grupos de usuarios, se debe fusionar el usuario actual con el grupo de usuarios más parecido: el problema radica en identificar cual es el grupo de usuarios más parecido. A continuación se exponen las distintas formas de hacerlo:
  - El de mejor encaje: el usuario más parecido será el de mejor función de encaje.
  - Función de error en el encaje: Se fusionara el usuario actual con cada uno de los 50 usuarios de la tabla y para cada una de estas fusiones se comprueba el encaje, es decir, se fusiona el usuario actual con un grupo y posteriormente se mira de nuevo el encaje del usuario actual con el grupo con que se fusionó, se almacenan los valores de encaje para cada grupo y se selecciona el mejor.
  - Función de error en la inferencia: Igual que en la opción anterior se encaja el usuario actual con todos los de la tabla y una vez encajado, se pregunta por las características del grupo con el que se encajó y se mira si coinciden con las del usuario actual, el grupo que mejor coincida será el elegido para la fusión. Para determinar cómo de parecidas son las características, se realizará la diferencia en valor absoluto.

- Función de pérdida de conocimiento: Se debe medir el conocimiento relativo y ver la pérdida de conocimiento relativo al encajar al usuario actual con cada grupo.
- Encaje relativo de un grupo consigo mismo: Al encajar al usuario actual con cada grupo, se debe mirar cómo perjudica al resto de grupos, es decir no solo considerar el encaje con un grupo sino la pérdida de conocimiento en otros grupos por este encaje. El grupo que menos perjudique al encajar, será el que se fusione con el usuario actual.

- Almacenamiento de información: Se deben mantener dos estructuras similares a las del módulo de encaje, en las que se almacenará la información de los dos grupos de usuarios que se van a fusionar. Tal y como se procedió para el caso del encaje la información de los usuarios de almacenará en memoria principal, en vez de en la base de datos.
- Realizar la fusión: El mecanismo se encarga también de llevar a cabo la fusión de los dos grupos. Para llevar a cabo esta tarea, cada una de las características de los grupos de usuarios a fusionar, se clasifican en los siguientes casos, que se exponen a continuación, junto con la formulación matemática de la fusión. Sean los conjuntos A y B, de las características de los dos grupos de usuarios a fusionar. Sean c(A), las características del conjunto A y sean c(B), las características del conjunto B. Así pues tenemos:

Para toda característica de c(A), identificamos las mismas posibilidades que para el caso del encaje:

- No existe c(B), serian *características disjuntas.*
- Existe c(B) y el valor de c(B), es el mismo que el valor de c(A), serían *características equivalentes.*
- Existe c(B) y el valor de c(B), es distinto que el valor de c(A), serían *características coincidentes.*

Según se cumplan las condiciones anteriores, se aplicará un caso u otro, en la fusión, de forma similar a como se procedía en el encaje. Se muestran las formulas matemáticas aplicadas a cada característica en la fusión del modelo. Se define popu1 y popu2 como las cardinalidades de las características A y B.

- Características disjuntas:
  - Modelo en [0,1]

- $Cert(C_k) = ((Z_k - 0.5 * popu1) / (popu1 + popu2)) + 0.5$

➢ Modelo en [-1,1]

- $Cert(C_k) = ((Z_k * popu1) / (popu1 + popu2))$

o Características coincidentes:

➢ Modelo en [0,1]

- $Cert(C_k) = ((Z_k - 0.5 * popu1) / (popu1 + popu2)) + 0.5$

➢ Modelo en [-1,1]

- $Cert(C_k) = ((Z_k * popu1) / (popu1 + popu2))$

o Características equivalentes:

➢ Modelos en [0,1] y [-1,1]

- $Cert(C_k) = ((Z_k * popu1) + (Z_j * popu2)) / (popu1 + popu2)$

- Retorno de la fusión: por último el subsistema de fusión, debe ser capaz de retornar, los resultados de la fusión, es decir el grupo único de usuarios fusionado.

  Como puede observarse tanto las formulas del encaje como las formulas de la fusión son equivalentes y congruentes para ambos rangos de valores de los modelos, es decir, en todos los casos las formulas en [-1,1] resultan de una conversión de las formulas en [0,1].

5.1.3. Subsistema de inferencia:

El subsistema de inferencia, permite preguntar al modelo, sobre un usuario e inferir nuevo conocimiento sobre este. Por tanto, para este módulo identificamos dos formas en las que se puede preguntar al modelo.

- Conociendo la característica: Se pregunta por su valor y su certeza
- Conociendo la característica y el valor: Se pregunta por su certeza

Para el modelo que nos ocupa, principalmente se realizará inferencia empleando la primera opción, es decir, siendo conocida la característica por la que se quiere preguntar, averiguar su valor y su certeza, a continuación desglosamos esta alternativa:

Tal y como se explicó, para el usuario actual conocida una característica, se pregunta por su valor y su certeza. Para realizar esta tarea, se plantean varias posibilidades:

- Primar la probabilidad

  Primar la probabilidad significa que se aplicarían las siguientes reglas:

  En primer lugar, se calculará el encaje del usuario actual con el resto de grupos, y para el de mejor encaje:

  - Si existe la característica, se devuelve
  - Si no existe la característica, se podría devolver desconocimiento o bien los datos del segundo o tercer grupo con que mejor encaje el usuario actual.

- Primar la certeza

  - Se devolvería siempre la característica de mayor certeza.

Por razones de fiabilidad y eficiencia del modelo, se prefiere la primera de las dos opciones, si bien posteriormente se analizara cuál de ellas se implementa.

5.1.4. Subsistema de estructura

El módulo de estructura, se encarga de la representación del modelo de usuario, así como de las funcionalidades propias de este. Por tanto, este subsistema es el encargado de la gestión del modelo en sí, de la estructura del modelo y de los procedimientos básicos del modelo. A continuación se detallan los procedimientos básicos del modelo:

- Procedimiento start: el modelo debe contar con un procedimiento capaz de iniciarlo.
- Procedimiento drop: se emplea para destruir la totalidad del modelo.
- Procedimiento reset: realiza un drop y posteriormente un start del modelo.

Estos procedimientos son ofrecidos por el modelo de usuario y son servicios hacia el exterior. Dentro de los servicios propios del modelo de usuario se han encontrando los tres siguientes que se detallan a continuación:

- Pedir información sobre el usuario actual en un contexto: este servicio se divide a su vez en dos nuevos servicios:

  - Adquisición de contexto: consiste en adquirir el contexto de un nuevo usuario, para ello se emplean los siguientes subservicios:

- Crear nuevo usuario: crea un nuevo usuario en el modelo.
- Añadir contexto: añade información del contexto del usuario al modelo.

- Preguntas: Tal y como se explico en el subsistema de inferencia, se pregunta sobre conocimiento inferido sobre el usuario actual.

- Aprender el contexto de un usuario: Permite al modelo aprender el contexto del usuario actual, esto se debe realizar cuando el usuario termina su sesión y todo su contexto es aprendido por el sistema.

  - Respuestas: existen tres tipos de respuestas posibles, estos se detallan a continuación:

- A priori: sin experiencia ni contexto sobre el usuario.
- Por defecto: con experiencia y sin contexto, es decir, respuesta estadística sin tener en cuenta al usuario actual.
- Computada: con experiencia y contexto, teniendo en cuenta al usuario actual.

5.1.5. Subsistema cliente

El subsistema cliente, es el encargado de gestionar el modelo de usuario (figura 03) se encarga de realizar los entrenamientos del modelo de usuario y enlazar el modelo de usuario, con la base de datos que contiene la información de origen. A continuación se expone el diagrama de casos de uso del modelo de usuario, con el fin de facilitar la comprensión del funcionamiento del sistema.

5.1.5.1. Diagrama de casos de uso del sistema

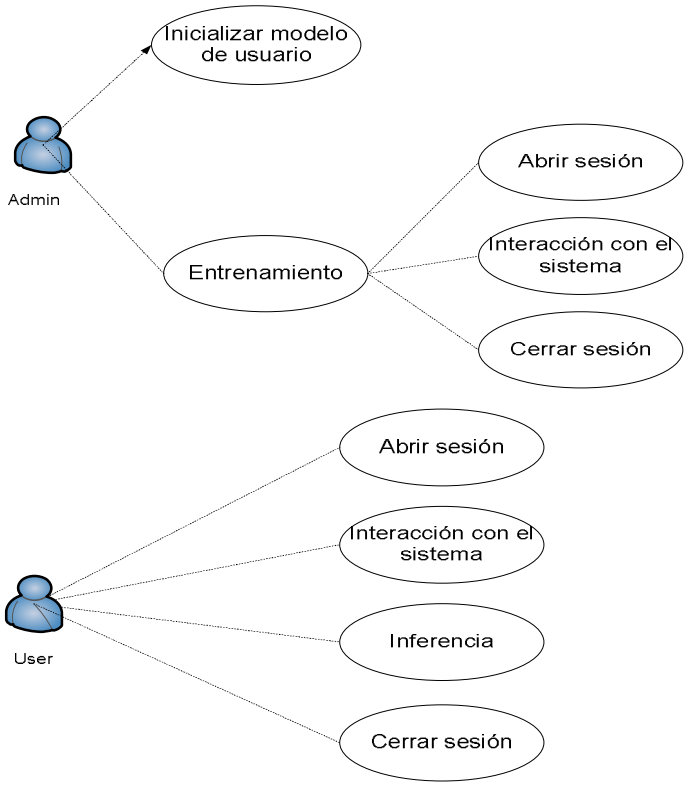


Ilustración 03. Diagrama de casos de uso

A continuación explicaremos brevemente el diagrama anterior, con el fin de analizar las funcionalidades del sistema. Como puede observarse en el diagrama, existen dos roles diferenciados, que interaccionan con el sistema, estos son el administrador y el usuario, cada uno de ellos puede llevar a cabo distintas funcionalidades, si bien existen ciertas funcionalidades que son compartidas por ambos.

Según el diagrama anterior, el administrador podrá llevar a cabo las funciones de inicializar el modelo de usuario, así como la de entrenar el modelo, mientras que el usuario podrá abrir su sesión, interaccionar con el sistema, inferir nuevo conocimiento y cerrar su sesión. Como puede observarse en el diagrama, la funcionalidad de entrenar el modelo llevada a cabo por el administrador, hace uso de funcionalidades propias del usuario como son abrir sesión, interacción con el sistema y cerrar sesión.

La siguiente tabla (tabla 01) detalla los casos de uso expuestos en la figura 03:

Identificador	Usuario	Nombre
CU-001	Administrador	Inicializar modelo de usuario
CU-002	Administrador	Entrenamiento
CU-003	Usuario/Administrador	Abrir sesión
CU-004	Usuario	Interacción con el sistema
CU-005	Usuario/Administrador	Inferencia
CU-006	Usuario/Administrador	Cerrar sesión

Tabla 01. Casos de uso

A continuación exponemos un diagrama de secuencia para cada uno de los casos de uso detallados anteriormente (tabla 01).

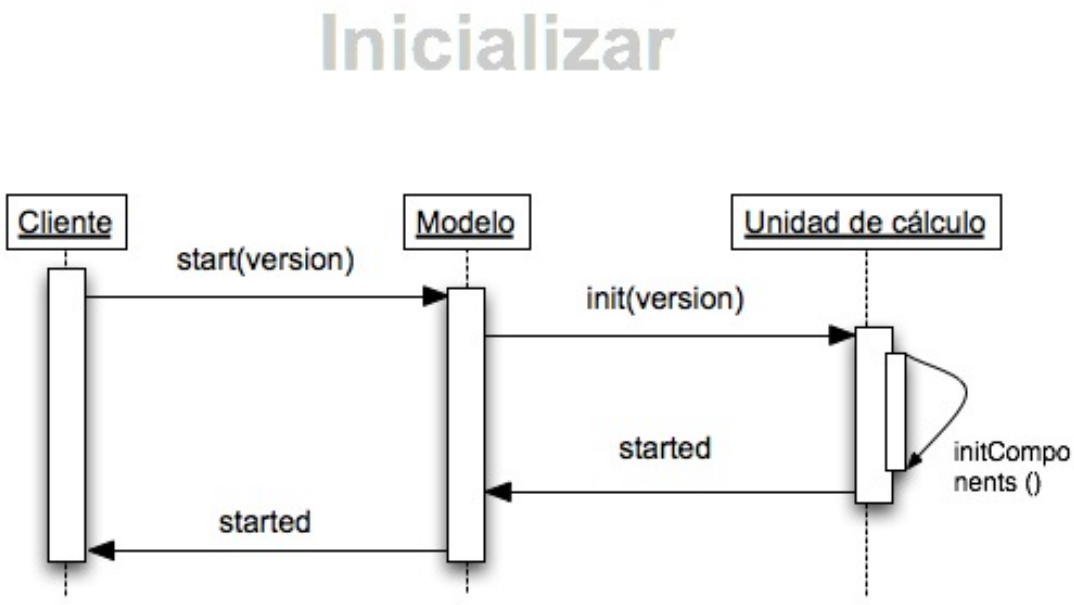


Ilustración 04. Diagrama de secuencia CU-001

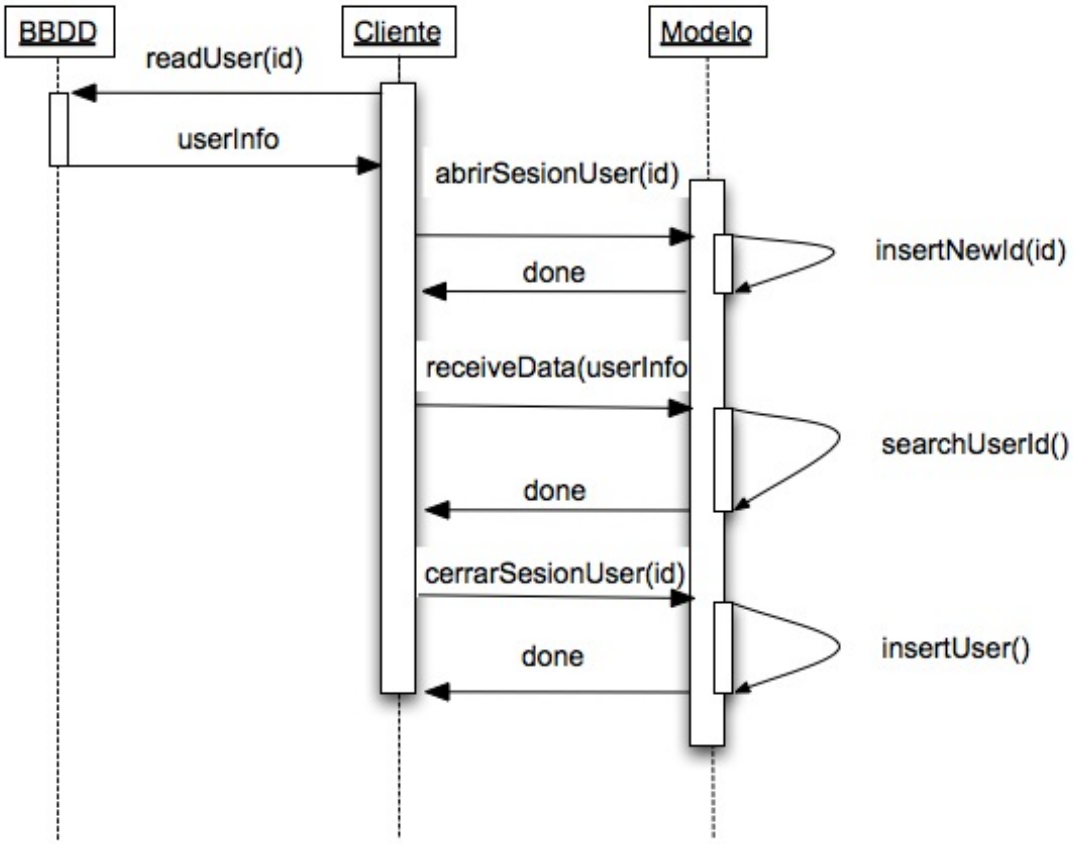


Ilustración 05. Diagrama de secuencia CU-002

La ilustración 05 expone otro diagrama de secuencia, para el caso de uso CU-002, teniendo en cuenta que tal y como ya se comentó, en el entrenamiento, puede darse el caso de que se produzcan fusiones, por exceder el volumen de información máximo del sistema.

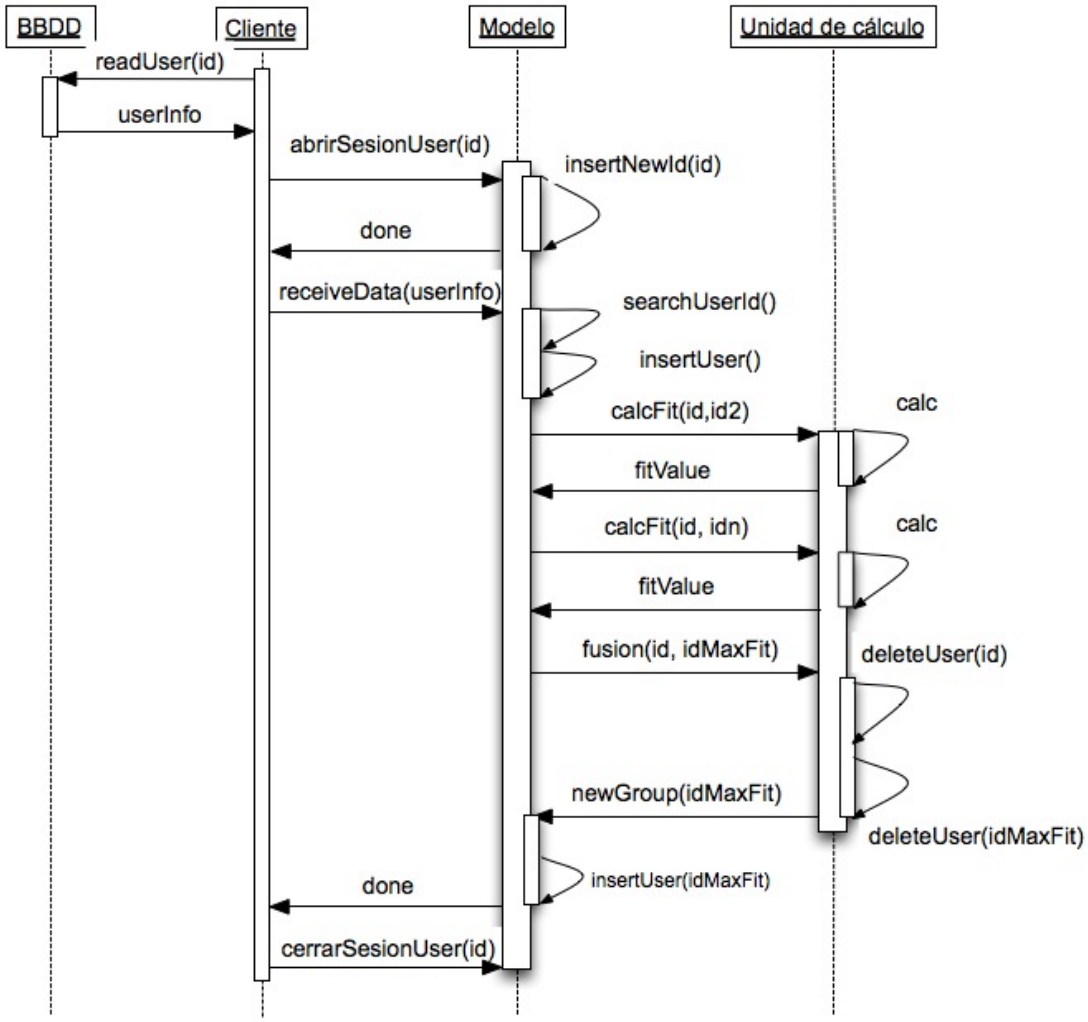


Ilustración 06. Diagrama de secuencia CU-002

Dada la complejidad del diagrama anterior, se procede a una breve explicación del mismo, con el fin de facilitar su comprensión.

En primer lugar, se puede observar en la ilustración 06 una línea de activación en el cliente, por tanto, el objeto cliente lee de la base de datos de origen los datos de un nuevo usuario. En el siguiente paso, el cliente solicita al modelo abrir una nueva sesión para un nuevo usuario cuyo identificador se le pasa al modelo, y a continuación el modelo creará el nuevo identificador temporal de usuario y avisará al objeto cliente de que terminó esta tarea. Seguidamente, el cliente envía al modelo la información del usuario que leyó de la base de

datos. De nuevo el modelo tiene el control o línea de activación, ya que buscará el identificador de usuario creado e insertará el nuevo usuario. A continuación, y puesto que este diagrama supone el caso de superar el nivel de información máximo del modelo, se debe producir una fusión de dos grupos de usuarios. Por lo tanto, a continuación el modelo solicitará a sus módulos embebidos que calculen el encaje del usuario actual con todos los grupos de usuarios. Finalmente, el grupo que mejor encaje se fusionará con el usuario actual.

Posteriormente, los módulos embebidos se encargarán de borrar del modelo los dos grupos fusionados, para pasarle al modelo la información del nuevo grupo, que será insertado. Finalmente, cuando se cierra la sesión del usuario actual, es cuando el modelo almacena de forma permanente, los datos del usuario.

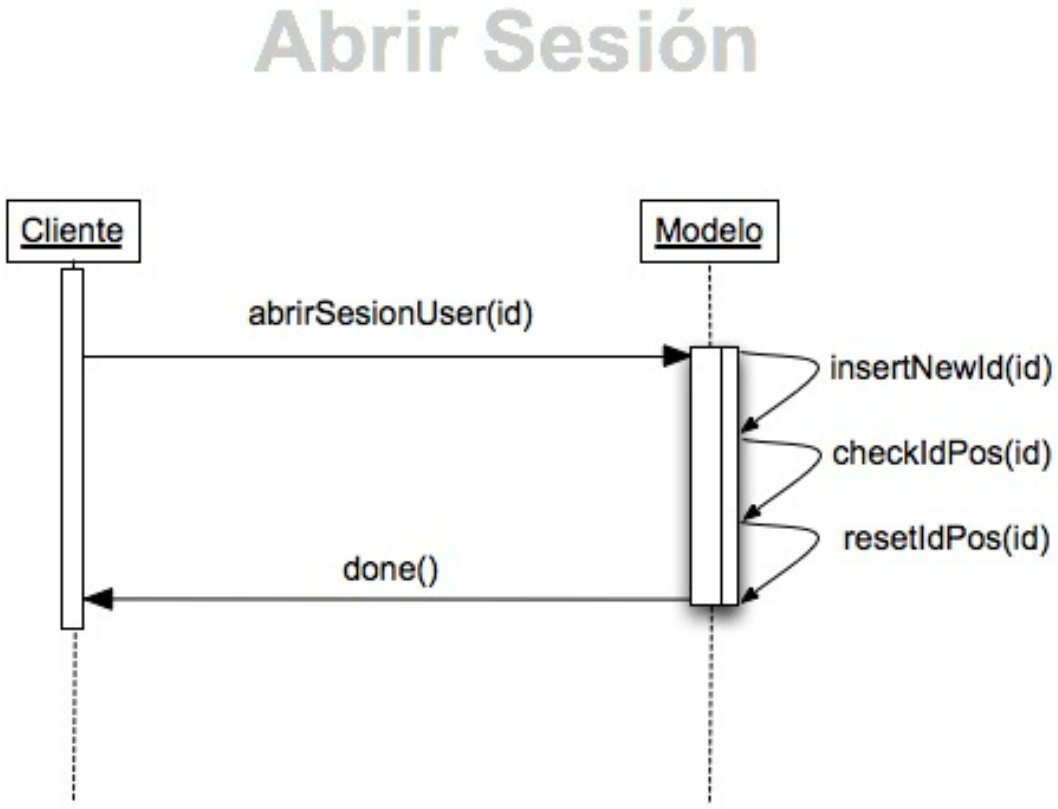


Ilustración 07. Diagrama de secuencia CU-003

A continuación, analizamos el diagrama de secuencia anterior (ilustración 07): Tal y como se vio en el caso anterior, el cliente solicita abrir una nueva sesión, para un nuevo usuario, el modelo, buscará por tanto la posición donde se insertará dicho identificador en una estructura temporal. Tras realizar la inserción, comprobará que efectivamente, esta fue exitosa y procederá

a realizar un reset del resto de datos de esa posición, con el fin de asignar a todos los datos de dicha posición de la tabla un valor por defecto. Este proceso anteriormente descrito es el que puede observarse en el diagrama de secuencia anterior.

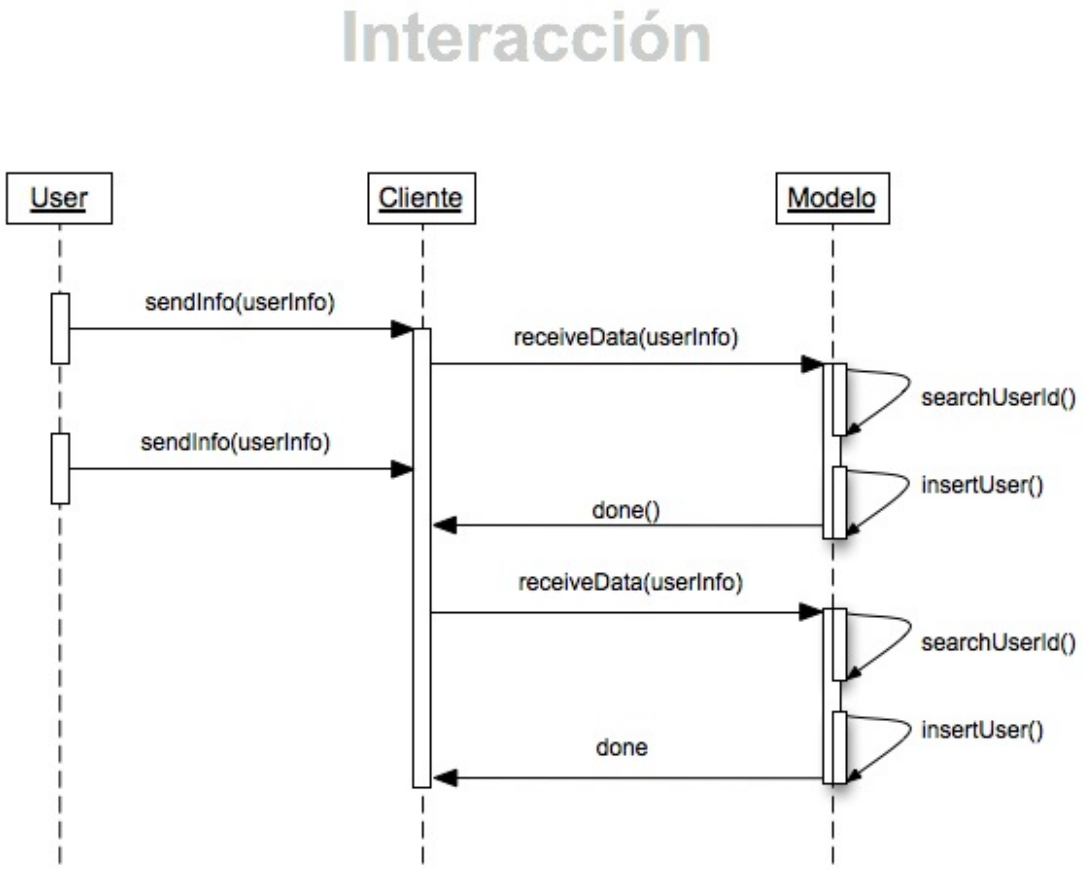


Ilustración 08. Diagrama de secuencia CU-004

Como puede observarse, el diagrama de secuencia del caso de uso CU-004 (ilustración 08) de interacción con el sistema, el usuario envía nueva información y el cliente la envía al modelo que la procesará y almacenará, de manera que se pueda inferir sobre dicha información.

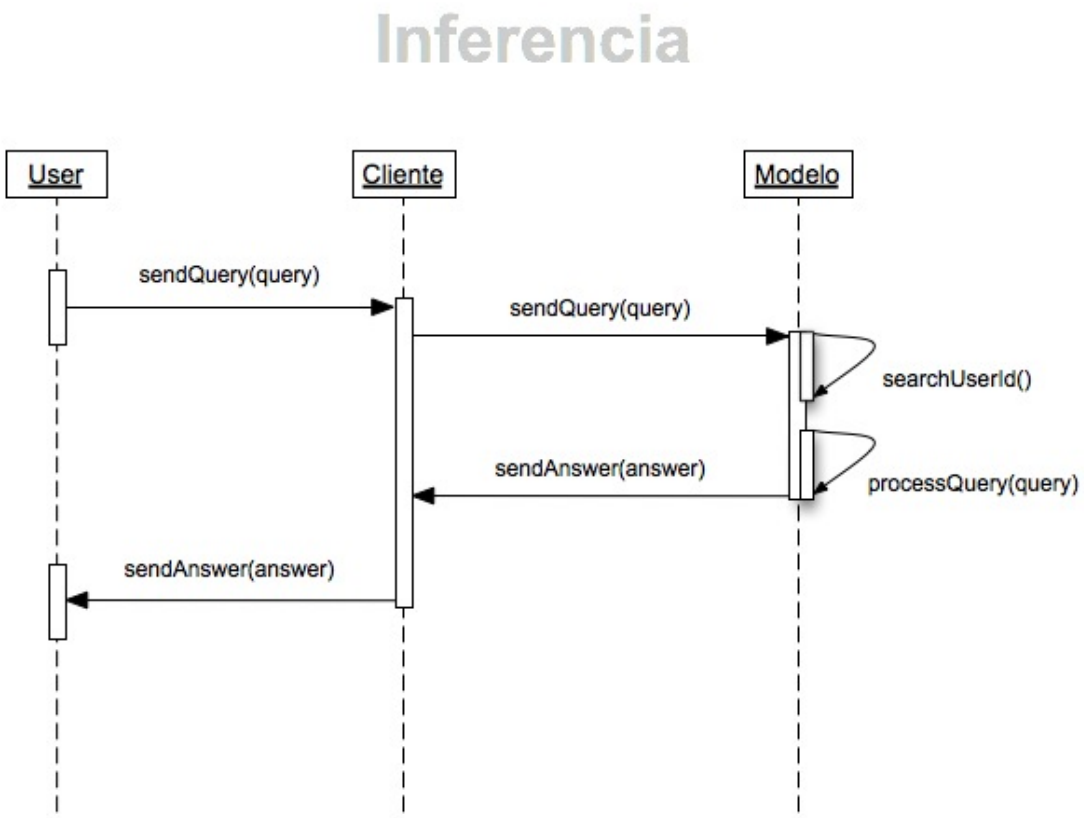


Ilustración 09. Diagrama de secuencia CU-005

El diagrama anterior muestra el caso de uso de inferencia (ilustración 09), en el que el usuario pregunta por una característica, y el modelo le responde a la pregunta. Como ya se vio anteriormente, el usuario puede preguntar por un valor y una certeza, o bien si el valor es conocido, preguntará solamente por una certeza.

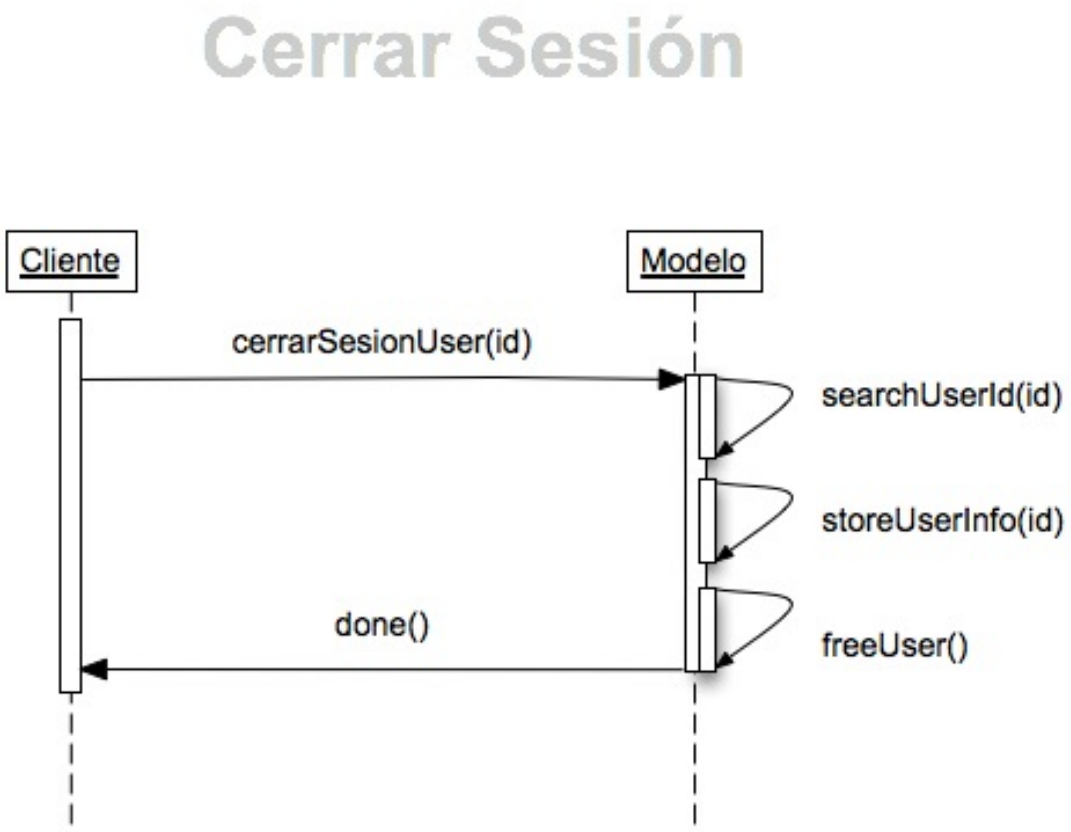


Ilustración 10. Diagrama de secuencia CU-006

Tal y como puede apreciarse en el diagrama anterior, cuando se cierra la sesión de un usuario, es cuando toda su información se vuelca de forma permanente al modelo y posteriormente se libera el espacio ocupado por dicha información.

5.2. Análisis y diseño de necesidades de información

En esta sección, se analizará la información que el sistema debe almacenar, con el fin de proporcionar la funcionalidad detallada en el capitulo anterior de la documentación. Así pues, partiendo de que el sistema cuenta con una base de datos que contiene los datos de origen, se obviará el almacenamiento de dichos datos; por otro lado, en lo que se refiere a la información propia del sistema, este debe almacenar la información referente al modelo de usuarios. Dentro de esta información podemos distinguir varios tipos de datos: los datos del modelo en sí mismo, y otros datos de control que permiten el buen funcionamiento del sistema. En esta sección se presentará un diagrama entidad relación que muestra las necesidades de información del sistema. Destacar que el objetivo, en lo que a necesidades de información se refiere, no es el de

encontrar una solución global, sino más bien una solución al caso concreto que nos concierne. Seguidamente, exponemos el diagrama entidad relación del modelo.

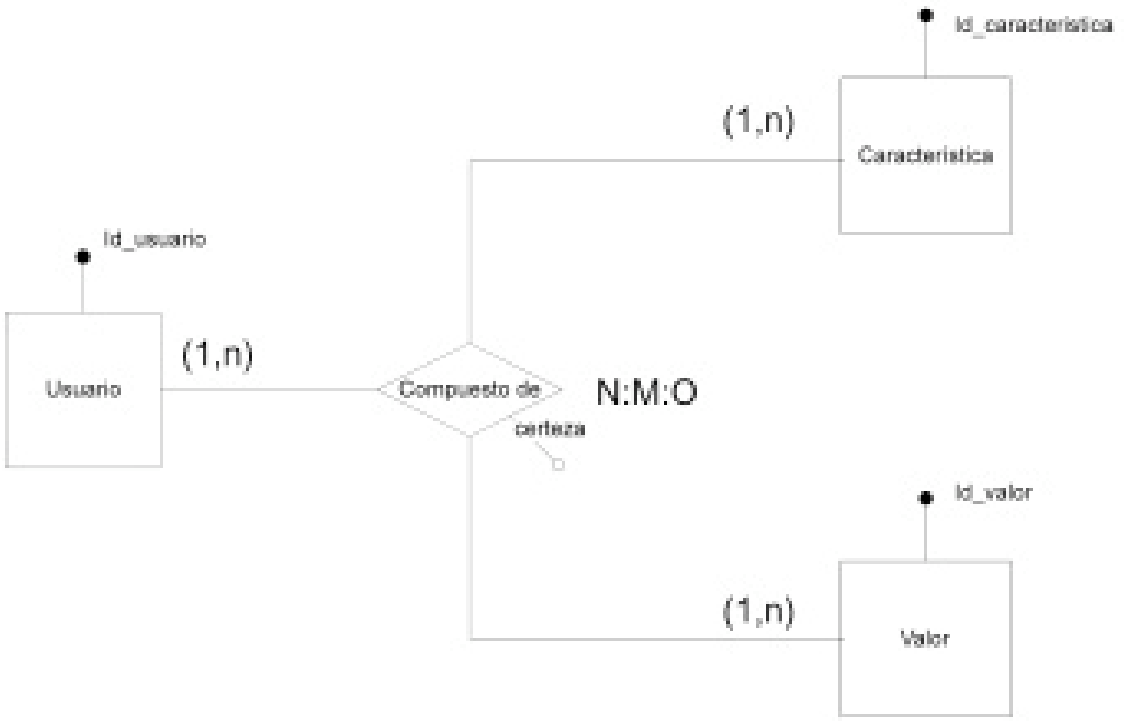


Ilustración 11. Diagrama entidad relación

Tal y como puede observarse en el diagrama anterior (ilustración 11), existe una relación ternaria entre las tres entidades que contiene el modelo, y cada una de estas entidades contiene un solo atributo que es clave principal. A continuación, analizaremos cada una de las entidades del modelo.

- Entidades:
  - Usuario: Almacena los usuarios obtenidos de la base de datos original y que estarán en el modelo.
  - Característica: Contiene todas las características que puede tener cualquiera de los usuarios de la entidad Usuario.
  - Valor: Guarda todos los valores que pueden tomar las características de los usuarios.
- Interrelaciones:

El esquema entidad relación, está compuesto por una sola interrelación ternaria, que relaciona las tres entidades analizadas anteriormente. Se trata de una interrelación con cardinalidad M:N:O. Por tratarse de una interrelación ternaria, se analizarán por separado cada una de las entidades participantes con respecto a las otras entidades, así pues se dejarán en cada caso fijas las cardinalidades de dos entidades y se comprobaran las combinaciones posibles para la entidad restante. Analizamos por tanto, las posibles combinaciones:

- Usuario – Característica: Si fijamos las entidades “Usuario” y “Característica”, tenemos que podemos contar con al menos un valor y máximo n valores distintos. Si no existiese valor, la característica no se almacenaría en el modelo, por tanto la cardinalidad es de 1:N.
- Característica – Valor: Para una característica y un valor, debe al menos existir un usuario, ya que de no ser así, la característica y el valor no tendrían sentido, ni aportarían conocimiento al modelo, por lo tanto la tupla anterior debe ir ligada al menos a un usuario. A su vez, la tupla anterior puede ir ligada a varios usuarios, por lo que de nuevo la cardinalidad es de 1:N.
- Usuario – Característica: Fijando la tupla, usuario y característica, observamos que pueden ocurrir varios valores para la tupla, y al menos debe existir uno, por lo que de forma análoga a como sucedía en los casos anteriores, la cardinalidad es de 1:N.

El análisis anterior es el que nos lleva a tener una interrelación de tipo N:M:O en el diagrama entidad relación anterior (ilustración 11).

Finalmente se expone el modelo relacional correspondiente al diagrama entidad relación anterior, con el fin de mostrar el diseño físico del sistema.

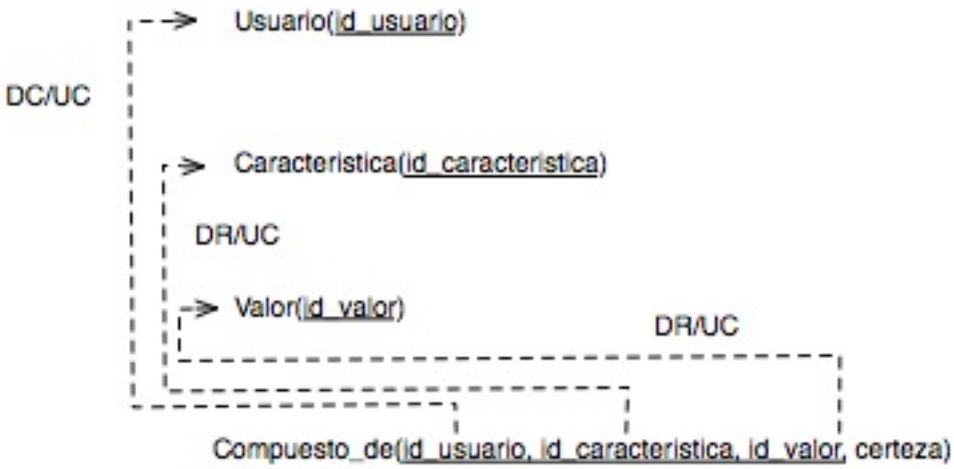


Ilustración 12. Modelo relacional

Tal y como se observa en la ilustración 12, el grafo relacional es una traducción del anterior diagrama entidad relación y recoge el diseño lógico que se empleará en la implementación del modelo.

A continuación y como se detalló anteriormente, se exponen las entidades auxiliares del modelo que este emplea para almacenar los datos de control necesarios para el funcionamiento del sistema. Como ya se dijo, dicho sistema debe guardar en otra entidad los datos necesarios para llevar a cabo los entrenamientos. El diagrama entidad relación es el siguiente (ilustración 13):

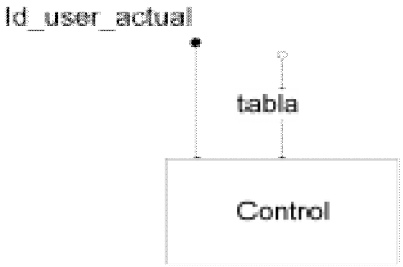


Ilustración 13. Entidad auxiliar

Tal y como muestra la ilustración 13, los datos de control se almacenan en una sola entidad, dicha entidad tiene como clave primaria el identificador del usuario actual, y se almacena también la estructura de la base de datos original en la que se encuentra dicho usuario.

5.3. Análisis y diseño físico

En este apartado se pretenden analizar las necesidades físicas del sistema, es decir, la distribución de cada uno de los módulos del sistema, con el fin de garantizar la eficiencia del sistema. A continuación se expone un diagrama con su diseño físico que ofrece las nociones necesarias para comprender las necesidades físicas del sistema.

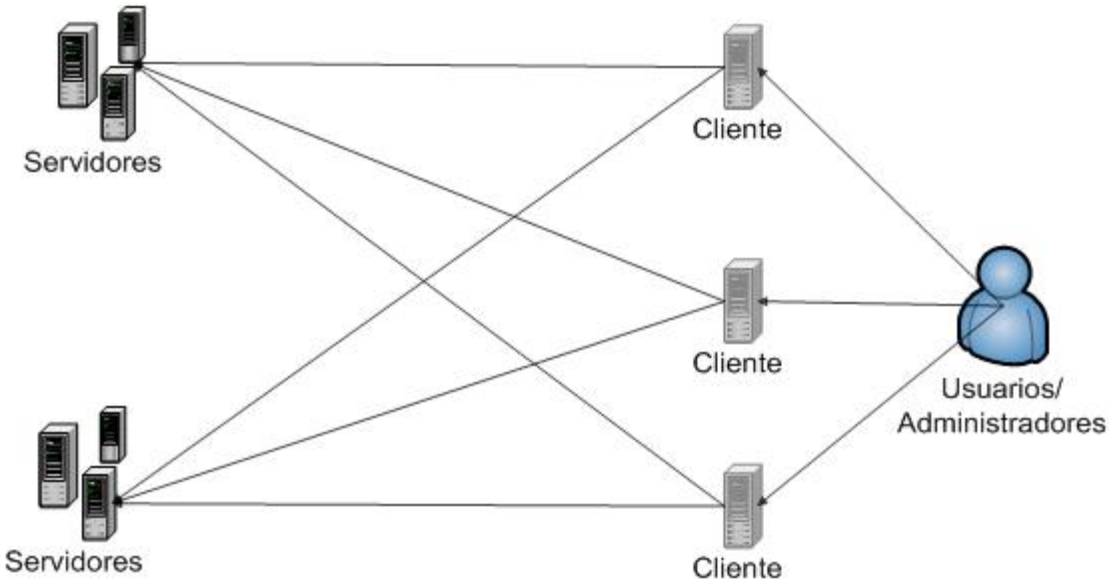


Ilustración 14. Diseño físico del sistema

Tal y como muestra el diagrama de la ilustración 14, el sistema contará con una o varias máquinas que actuarán como clientes y que contendrán el subsistema cliente explicado anteriormente en la presente sección. Dichos clientes serán el punto de acceso de los administradores y en todo caso de los usuarios al sistema. Estos clientes se comunicarán con los servidores disponibles en los que se distribuyen el resto de módulos del sistema. Sin embargo, este apartado, no se desea entrar en detalles de comunicaciones, ya que este aspecto es inherentemente dependiente de la tecnología, por lo tanto las comunicaciones del sistema serán tratadas en el capítulo de implementación del presente documento. Por tanto, pasamos al análisis de los servidores necesarios para el sistema, a saber, serán necesarios uno o más servidores de bases de datos, en los que se almacenarán los datos de origen con que se entrenará el sistema, así como el modelo. Por otra parte serán necesarios uno o más servidores para dar soporte a las unidades de cálculo que maneja el sistema.

5.4. Conclusiones

A lo largo del capítulo de análisis y diseño se han expuesto posibles soluciones a todos los problemas y subproblemas del caso que nos ocupa, a su vez, se han introducido las soluciones de diseño adoptadas, tanto en el aspecto funcional como en el de necesidades de información y como en el apartado físico. Por lo tanto, toda la información y el análisis llevado a cabo en la sección actual serán el punto de partida de la siguiente sección o implementación del sistema. Así mismo, puede afirmarse que el seguimiento del análisis y diseño realizado en la fase de implementación serán claves para el correcto desarrollo del sistema.

6. Implementación

Tras el proceso de análisis y diseño, se procede a llevar a cabo la implementación del sistema, por lo tanto, en la presente sección, se va a describir cómo se aplicarán todos los pasos descritos en la fase de diseño del sistema durante el proceso de implementación. Se realizarán comprobaciones periódicas, con el fin de asegurar que el diseño propuesto se está llevando a cabo, lo cual es clave para la correcta implementación del sistema. Así mismo, en este apartado se introducirán todas las tecnologías empleadas en el desarrollo del sistema, así como las decisiones propias de implementación.

Con objeto de facilitar al lector el seguimiento del capítulo, así como la comprensión del proceso de implementación llevado a cabo durante el paso del diseño a la implementación, se explicarán y analizarán todos los pasos que se llevados a cabo para la consecución de la codificación del sistema cuyo diseño se ha detallado en capítulo anterior.

De forma análoga a como se procedió en el capítulo de análisis y diseño, esta sección se descompondrá en varios apartados en los que se tratarán aspectos funcionales, de necesidades de información y físicos. En este sentido, se expondrán diagramas de clases, diagramas relacionales y diagramas de arquitectura física del sistema. El capítulo de implementación, también detallará en lo que se refiere a la arquitectura física del sistema, las especificaciones de las maquinas en las que se instalará el sistema y se entrenará, con el fin de ofrecer una visión de los requisitos o necesidades recomendables para este cometido.

Así pues, tras el capítulo de implementación, se tendrán nociones suficientes en cuanto a la funcionalidad del sistema así como a los detalles físicos de este. En este apartado se introducirán también las tecnologías que han intervenido en el desarrollo del sistema, como puedan ser los lenguajes de programación Java, o PL/SQL, así como el gestor de bases de datos Oracle. De forma análoga a todo lo detallado anteriormente, todos los problemas encontrados durante la fase de implementación del sistema serán analizados en la presente sección del documento.

6.1. Aspectos funcionales de implementación

Tal y como se analizó en el capítulo de análisis y diseño, el sistema propuesto se divide en varios módulos, módulo de encaje, módulo de fusión, módulo de inferencia, módulo de estructura y módulo cliente. A continuación se exponen los datos referentes a la implementación de cada uno de los módulos del sistema.

6.1.1. Implementación subsistema de encaje

A continuación se describen los aspectos de implementación del subsistema encargado de realizar las tareas de encaje en el modelo.

El subsistema de encaje fue implementado en Java y realiza conexiones con el modelo mediante secure socket, así pues, cada vez que el modelo necesita realizar operaciones de encaje, abrirá una conexión secure socket con el subsistema de encaje.

6.1.1.1. Almacenamiento

El módulo de encaje, es capaz de realizar el encaje entre dos usuarios o grupos de usuarios, para ello, y tal y como se vio en el capítulo de diseño, este módulo debe almacenar los datos de los grupos sobre los que se calculara el encaje. Para el almacenamiento de la información, se deben considerar que todo grupo de usuarios divide su información en tuplas formadas por “identificador de usuario”, “característica”, “valor” y “certeza”, tal y como se vio en el apartado de diseño de esta memoria. Por otro lado, los datos de los grupos son volátiles y cambian de una operación a otra, por tanto existen dos clases Java encargadas del almacenamiento de los datos de los usuarios. Las clases encargadas de realizar el almacenamiento de la información de los usuarios, son las clases “UserConceptualization” y “UserFeature”. A continuación se expone el diagrama de clases de ambas clases Java, con el fin de comprender como se almacenan los datos de los usuarios antes de realizar la operación de encaje.

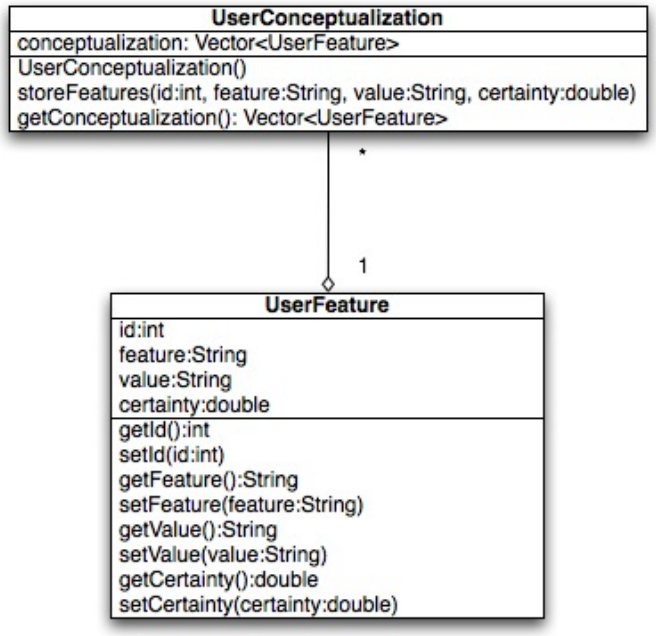


Ilustración 15. Diagrama de clases almacenamiento encaje

Como puede observarse, las dos clases anteriores son una abstracción del concepto de usuario con el que trabajará el modelo. Así pues, la clase "UserFeature" es una abstracción de un usuario con su información, mientras que la clase "UserConceptualization" permite almacenar varios datos de uno o varios usuarios. A continuación se detallan las funciones de los métodos de las clases anteriores.

6.1.1.1.1. UserConceptualization

Almacena en memoria la información de los usuarios para las operaciones del módulo de encaje. Los principales métodos de esta clase son:

- UserConceptualization(): Es el constructor de la clase y se encarga de crear los objetos de tipo UserConceptualization.
- storeFeatures(id, String, String, double): Método que permite almacenar tuplas del tipo identificador, característica, valor y certeza en el objeto de tipo UserConceptualization.
- getConceptualization(): Devuelve la abstracción de un usuario.

6.1.1.1.2. UserFeature

Clase encargada de la gestión y manipulación de los datos de los usuarios almacenados por la clase UserConceptualization. Los métodos principales de esta clase se describen a continuación:

- getId(): Devuelve el identificador de usuario.
- setId(int): Asigna un identificador de usuario.
- getFeature(): Devuelve una característica.
- setFeature(String): Asigna una característica.
- getValue(): Devuelve el valor de una característica.
- setValue(String): Asigna un valor a una característica.
- getCertainty(): Devuelve la certeza de un par característica valor.
- setCertainty(double): Asigna una certeza a un par característica valor.

6.1.1.2. Clasificación y encaje

La segunda tarea que debe realizar el módulo de encaje es la categorización de la información de los usuarios, para esta tarea se deben clasificar las tuplas “identificador de usuario”, “característica”, “valor”, “certeza” en una de las tres categorías siguientes, características disjuntas, coincidentes o equivalentes. Esta tarea se lleva a cabo junto con la tarea de encaje, tras el almacenamiento de la información de los usuarios. La clase encargada de esta tarea es la clase FitModel, junto con las clases UserConceptualization, UserFeature y PairFeatureValue. A continuación se expone el diagrama de las clases encargadas de estas tareas.

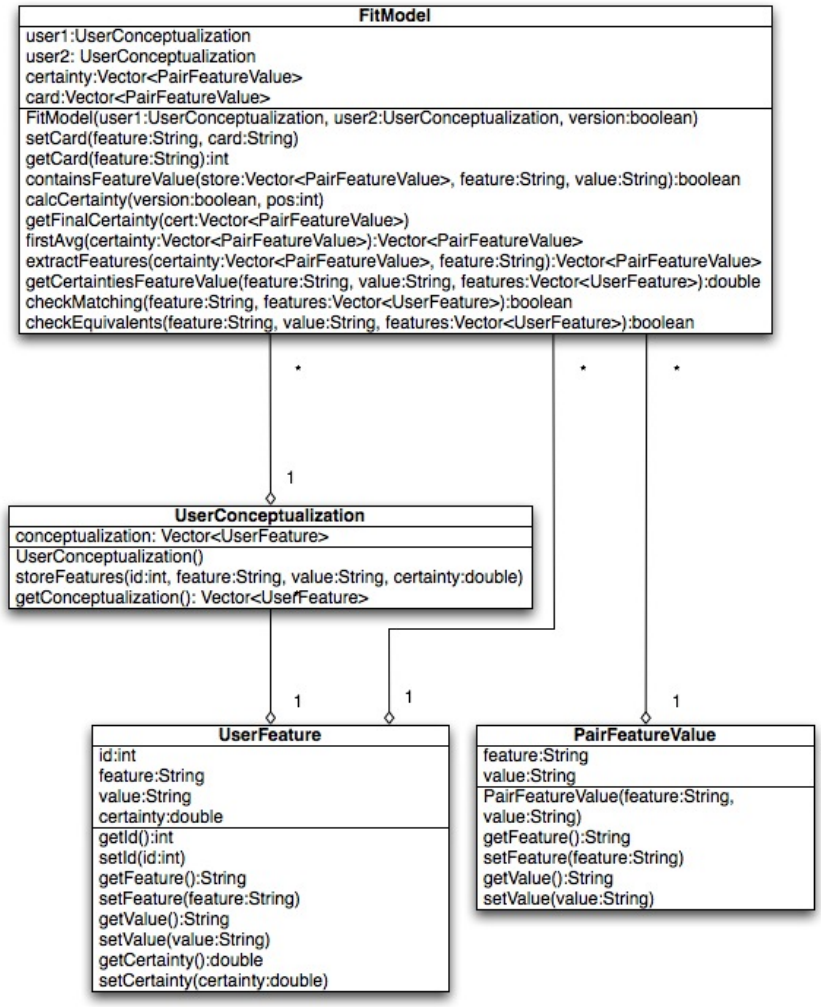


Ilustración 16. Diagrama de clases encaje

6.1.1.2.1. FitModel

A continuación se analizan los principales atributos y métodos del diagrama anterior. La clase FitModel, es la encargada de realizar la categorización de los datos de los usuarios, así como de realizar el cálculo del encaje para los grupos de usuarios dados. Esta clase cuenta con dos atributos de tipo UserConceptualization que son user1 y user2, por lo que está preparada para calcular el encaje entre dos grupos de usuarios. Estos atributos representan la abstracción de la información de los grupos sobre los que se realizara el encaje. Los dos siguientes atributos, certainty y card, son objetos de tipo PairFeatureValue, por lo que están compuestos de tuplas característica valor y se encargan de almacenar las certezas de los encajes y las cardinalidades de cada característica en el dominio respectivamente. Las certezas se irán calculando durante el proceso de encaje, mientras que las cardinalidades del dominio son fijas y conocidas a priori.

En lo que se refiere a los métodos de la clase FitModel, a continuación se detallan las principales funciones de cada uno de ellos.

- FitModel(UserConceptualization, UserConceptualization, boolean): Constructor de clase que crea los objetos de tipo FitModel.
- setCard(String, String): Almacena una característica con su cardinalidad correspondiente.
- getCard(String): Devuelve una característica.
- containsFeatureValue(Vector<PairFeatureValue>, String, String): Comprueba si un Vector de pares (característica, valor) contiene un determinado par (característica, valor).
- calcCertainty(boolean, int): Se encarga de los cálculos del encaje y será explicado con mayor nivel de detalle a continuación.
- getFinalCertainty(Vector<PairFeatureValue>): Ajusta las formulas del encaje para que sean congruentes mediante un factor de corrección que en este caso es dividir entre dos y sumar 0.5.
- firstAvg(Vector<PairFeatureValue>): Realiza la media aritmética de las certezas calculadas
- extractFeatures(Vector<PairFeatureValue>, String): Permite obtener los pares característica valor de un usuario que cumplan que el nombre de la característica sea igual a uno dado.
- getCertaintiesFeatureValue(String, String, Vector<UserFeature>): Devuelve la certeza de un par (caracteristica valor).
- checkMatching(String, Vector<UserFeature>): Comprueba si una característica es disjunta.
- checkEquivalents(String, String, Vector<UserFeature>): Comprueba si una característica es equivalente.

A continuación se explica el funcionamiento del método calcCertainty que es el más importante de la clase FitModel y es el encargado de realizar los cálculos del encaje para los dos grupos de usuarios almacenados, se encarga por tanto también de la categorización de la información de los usuarios, es decir, en clasificar cada una de las características de los usuarios en uno de los tres grupos posibles, características disjuntas, coincidentes o equivalentes. Una vez el método va clasificando cada una de las características de los usuarios, éste debe aplicar una función matemática u otra a la característica, en función del grupo en que esta se encuadro. Así pues, el funcionamiento del método es sencillo, puesto que recorre toda la información del

primer grupo de usuarios y para cada característica decide si esta es disjunta, coincidente o equivalente con relación al segundo grupo de usuarios, y posteriormente aplica una función matemática a dicha característica según la categoría de esta. Puesto que las formulaciones matemáticas aplicadas en cada caso ya se detallaron en el apartado de diseño del presente documento se remite al lector a dicha sección.

Por último, el método firstAvg se encarga de realizar las medias de las certezas calculadas durante el proceso de encaje y mediante las formulaciones planteadas con anterioridad. Este método realiza en ambos casos, [0,1] y [-1,1], medias aritméticas. Además, y de la misma forma, al resultado obtenido se le aplica un factor de corrección, que es el que permite asegurar la congruencia y equivalencia entre las formulas del dominio [0,1] y las del dominio [-1,1]. Por lo tanto, se puede considerar un factor de conversión que en este caso es dividir la certeza final entre dos y sumarle 0.5. De esta forma se puede concluir que las formulas de un dominio y otro serán equivalentes, lo que supone que las únicas diferencias en el encaje entre ambos modelos serán por los redondeos que intuitivamente favorecen ligeramente al modelo en [-1,1], ya que este formato aprovecha un bit más de representación que el modelo en [0,1].

La última tarea de la que debe encargarse el módulo de encaje es la de retornar el valor del encaje calculado al módulo Pl/Sql correspondiente. Esta función se realiza automáticamente en la llamada a las clases Java que realizan los módulos Pl/Sql, por lo que se analizará más adelante cuando se exponga la estructura de dichos paquetes Pl/Sql.

6.1.2. Implementación subsistema de fusión

En la presente sección se exponen los aspectos de implementación del módulo de fusión de los modelos desarrollados, dicho módulo, teniendo en cuenta su complejidad, fue implementado en Java. De la misma forma que el módulo de encaje, el módulo de fusión realiza conexiones mediante secure socket a los módulos Pl/Sql encargados de la estructura y funcionamiento del modelo.

6.1.2.1. Fusión

El módulo de fusión se encarga de, dados dos grupos de usuarios, fusionar dichos grupos en uno, de manera que el número de grupos de usuarios almacenados en el modelo, se mantenga constante y en un límite establecido que en nuestro caso se fijo a cincuenta usuarios. Tal y como se indico en el capítulo de análisis y diseño, la primera tarea del módulo de fusión es seleccionar el grupo de usuarios más parecido y con el que se realizara la fusión. Para llevar a

cabo esta tarea, se identificaron varias alternativas y finalmente se decidió adoptar la primera de ellas en la que se decide el usuario a fusionar en función del encaje. Los motivos por los cuales se decidió implementar esta alternativa en vez de las otras propuestas, es porque esta opción es la que proporciona a priori mejores resultados en lo que la eficiencia se refiere. Por lo tanto, el proceso de decisión del usuario a seleccionar se realiza mediante el módulo de encaje, es decir, se calcula el encaje del usuario actual con todos los grupos de usuarios almacenados en el modelo, y se fusiona con el de mejor encaje, por lo tanto, las clases que intervienen en este proceso serán las mismas que las explicadas anteriormente en el módulo de encaje. De igual forma sucede con la parte del almacenamiento de los grupos de usuarios a fusionar para los que se emplean las mismas clases que en el caso del encaje. Por último, la tarea de fusión llevada a cabo por el módulo de fusión, se realiza mediante las siguientes clases.

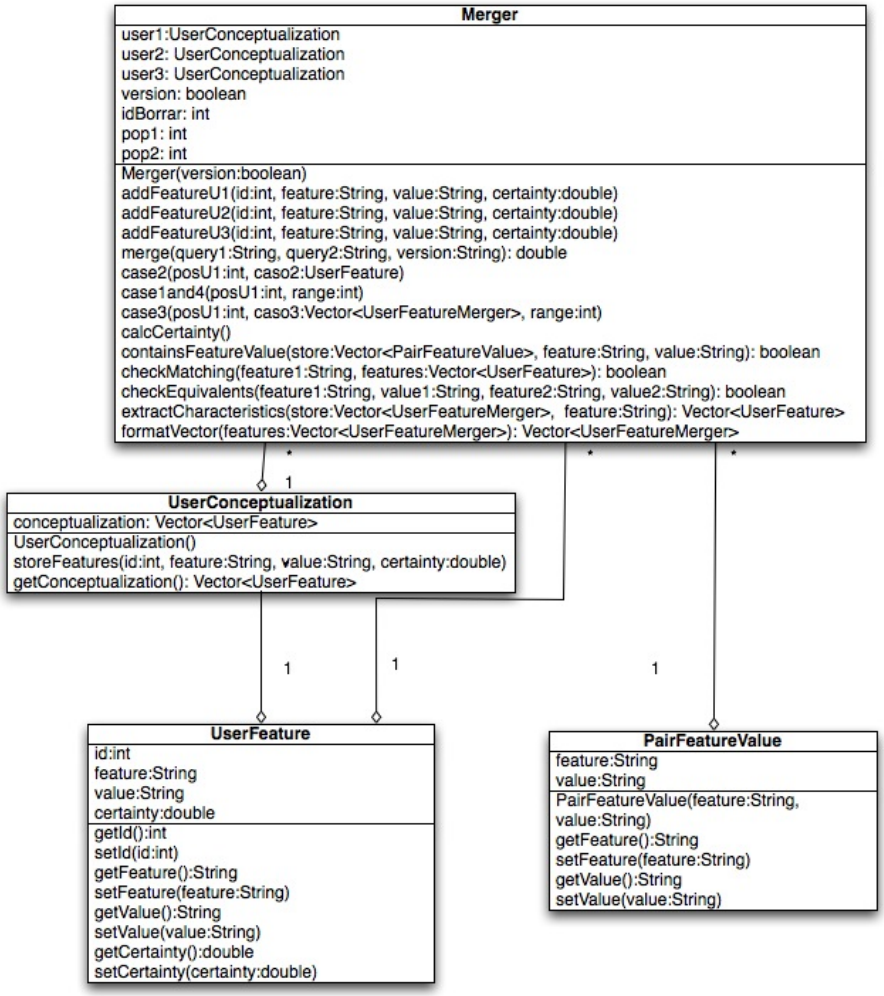


Ilustración 17. Diagrama de clases fusión

A continuación se analizan los principales atributos y métodos del diagrama de clases anteriormente expuesto. El diagrama expuesto muestra la totalidad de las clases que intervienen en el módulo de fusión, es decir, incluye aquellas clases que se encargan del almacenamiento de la información de los usuarios aunque éstas son las mismas que las empleadas para el módulo de encaje y fueron analizadas en el capitulo anterior de la presente documentación. Por lo tanto, a continuación nos centraremos en la clase Merger que es la encargada de realizar la fusión de dos grupos en uno solo.

6.1.2.1.1. Merger

Comenzando por los atributos de la clase Merger, tenemos tres atributos del tipo UserConceptualization, los dos primeros atributos se emplean para almacenar la información de los grupos de usuarios que fusionaran, el tercero de los usuarios se emplea para almacenar el grupo resultante de la fusión que posteriormente se insertará en la base de datos. El atributo versión, nos indica si se trata de un modelo que maneje valores en rango [0,1] o [-1,1] así pues si dicho atributo toma el valor “true”, se trata de un modelo en [0,1] y en caso contrario se trata de un modelo en [-1,1]. El atributo idBorrar, almacenará el identificador que posee en la base de datos el grupo de usuario con el que se fusiona, de esta manera luego se podrá borrar dicho grupo e insertar el nuevo grupo (el fusionado) con ese mismo identificador. Por último los atributos pop1 y pop2 almacenarán las poblaciones de los grupos de usuario a fusionar ya que dichas poblaciones intervendrán en los cálculos de la fusión. Ambas poblaciones se almacenan en memoria, ya que se considera más eficiente recuperarlas de memoria que de la base de datos ya que intervendrán en varios cálculos. Los métodos de la clase Merger tienen la siguiente funcionalidad.

- Merger(boolean): Constructor que crea los objetos de tipo Merger y asigna el rango del modelo en [0,1] si recibe “true” o en [-1,1] si recibe “false”.
- addFeatureU1(int, String, String, double): Permite almacenar la información del primer grupo que interviene en la fusión.
- addFeatureU2(int, String, String, double): Permite almacenar la información del segundo grupo que interviene en la fusión.
- addFeatureU3(int, String, String, double): Almacena el grupo resultante de la fusión, que será insertado en la base de datos del modelo.
- merge(String, String, String): Punto de acceso por parte de los módulos Pl/Sql al módulo de fusión.
- case2(int, UserFeature): Función aplicada a características equivalentes.

- case1and4(int, int): Función aplicada a características disjuntas.
- case3(int, Vector<UserFeature>, int): Funcion aplicada a características coincidentes.
- calcCertainty(): Método que realiza la fusión, su algoritmo se expone a continuación.
- containsFeatureValue(Vector<PairFeatureValue>, String, String): Comprueba si el Vector contiene la característica y el valor.
- checkMatching(String, Vector<UserFeature>): Comprueba si la característica dada en el primer parámetro es disjunta.
- checkEquivalents(String, String, String, String): Comprueba si dos características son equivalentes.
- extractCharacteristics(Vector<UserFeature>, String): Obtiene del Vector de entrada las tuplas cuya característica se llama igual que el parámetro de tipo String de entrada.
- formatVector(Vector<UserFeature>): Formatea un Vector agrupando las características del mismo nombre.

A continuación se expone el algoritmo empleado para llevar a cabo el proceso de fusión. Para realizar este algoritmo disponemos de dos conjuntos de características, las pertenecientes a cada grupo de usuarios, así pues en primer lugar se debe realizar la categorización de las características. Sean C(A) el conjunto de características de uno de los usuarios y C(B) el conjunto de características del otro usuario.

```
Tomamos todas las C(A) LOOP
 SI NO existe C(B) → caso I NEXT
        ELSE
                SI existe C(B) se toman {(Vi, Zi)} en C(A) LOOP
                        SI Vi existe en C(B) → caso II NEXT
                        SI Vi no existe en C(B) → caso III NEXT
                END LOOP
Tomamos {(Vj, Zj)} en C(B) LOOP
                        SI Vj existe en C(A) → no hacer nada NEXT
                        SI Vj no existe en C(A) → caso III NEXT
                END LOOP
        END IF
END LOOP
Tomamos todas las C(B) LOOP
        SI NO existe C(A) → caso I NEXT
        ELSE no hacer nada
        END IF
END LOOP
```

Ilustración 18. Algoritmo de fusión

A continuación se expone la explicación del algoritmo anterior, con el fin de entender el proceso realizado para la fusión de dos grupos de usuarios en el modelo de usuarios. En primer lugar tal y como se expuso anteriormente, la categorización de las características se realiza en los tres grupos siguientes, disjuntas, coincidentes y equivalentes, en el caso del algoritmo se identificaron tres casos distintos que se corresponden uno a uno con los tres tipos de características existentes. La correspondencia es la siguiente:

- Caso I equivale a características disjuntas
- Caso II equivale a características equivalentes
- Caso III equivale a características coincidentes

Así pues, el algoritmo comienza recorriendo las características de uno de los usuarios y categorizando cada una de ellas, es decir, aplicando un caso u otro según las condiciones que cumple dicha característica. Finalmente, se recorren las características del otro usuario con el fin de recoger también sus características disjuntas y que por tanto no se encuentran entre las características del primer usuario. Por tanto este módulo emplea las

funciones matemáticas expuestas en la sección 5.1. del presente documento y las aplica según la categoría de cada característica.

6.1.3. Implementación subsistema de inferencia

De las posibilidades que se expusieron en el apartado 5.1.3 finalmente se implementaron ambas, con el fin de poder comparar ambas y ver cuál de ellas se comporta mejor para el dominio de los datos. A continuación se expone el diagrama de clases del subsistema de inferencia.

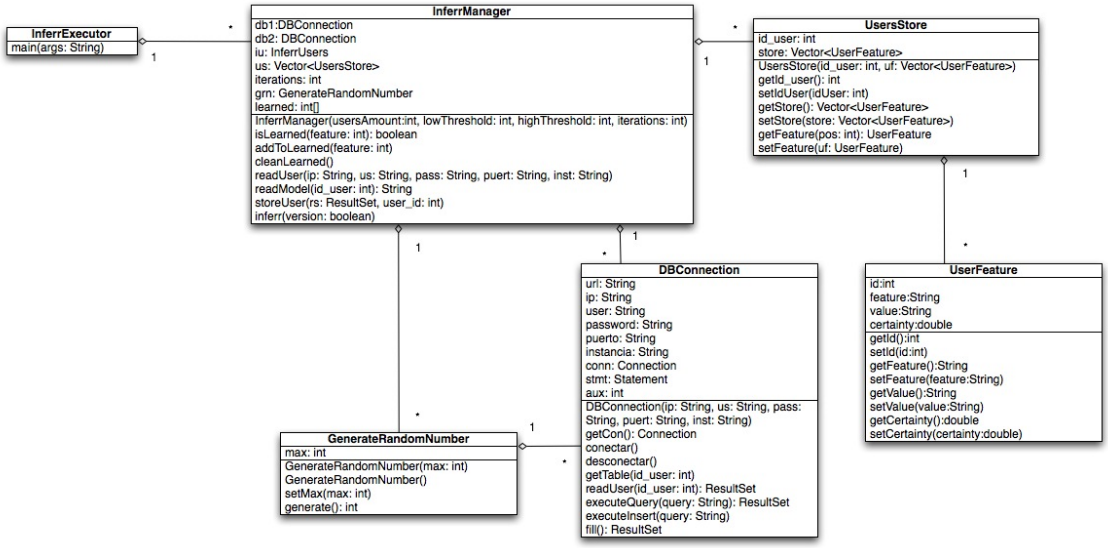


Ilustración 19. Diagrama de clases inferencia

En lo que se refiere al diagrama de clases anterior, éste modela el subsistema de inferencia del modelo de usuario, permite por lo tanto inferir conocimiento sobre el modelo. A continuación se expone una explicación detallada de cada una de las clases que lo componen.

6.1.3.1. InferrExecutor

Clase encargada de comenzar la ejecución del módulo de inferencia. Realiza la llamada a la clase InferrManager encargada de gestionar el proceso de inferencia.

- main(String): Método main que es el punto de inicio de las ejecuciones del subsistema de inferencia.

6.1.3.2. InferrManager

Clase encargada de gestionar y llevar a cabo el proceso de inferencia sobre el modelo de usuario. Cuenta con dos objetos de tipo DBConnection que se encargan de realizar las conexiones a las bases de datos del modelo de usuario y al corpus de usuarios. El atributo de tipo Vector<UserStore> permite almacenar los usuarios con los que se realiza la inferencia. Los tres últimos atributos de clase respectivamente permiten almacenar el número de iteraciones o características aprendidas para cada usuario durante la inferencia, generar números aleatorios para seleccionar los usuarios y almacenar en memoria las características del usuario que ya son conocidas para el modelo. En lo que se refiere a los métodos de la clase:

- InferrManager(int, int, int, int): Constructor de la clase InferrManager que permite crear objetos de dicha clase.
- isLearned(int): Permite conocer si una característica de un usuario es conocida o no por el modelo.
- addToLearned(int): Añade una característica al conjunto de características aprendidas por el modelo.
- cleanlearned(): Vacía el contenedor de características aprendidas por el modelo.
- readUser(String, String, String, String, String): Permite realizar una conexión a la base de datos que contiene el corpus de usuarios y leer un usuario.
- readModel(int): Lee un usuario de la base de conocimiento del modelo.
- storeUser(ResultSet, int) : Almacena un nuevo usuario para realizar inferencia sobre el modelo.
- Inferr(boolean): Realiza la inferencia sobre la base de conocimiento, recibe un parámetro que indica si se trata de un modelo con tratamiento de datos en rango [0,1] o en [-1,1].

6.1.3.3. GenerateRandomNumber

Clase que se encarga de la gestión de números aleatorios para seleccionar aleatoriamente los usuarios con los que se realiza la evaluación del modelo. La clase cuenta con un único atributo de clase que indica el valor máximo que puede tomar el número aleatorio generado. Los métodos de la clase GenerateRandomNumber son los siguientes:

- GenerateRandomNumber(int): Constructor de clase que recibe el máximo que se asignará al atributo de la clase.
- GenerateRandomNumber(): Constructor de clase sin parámetros.

- setMax(int): Asigna un máximo para la creación de números aleatorios.
- generate(): Genera un numero aleatorio entre cero y max.

6.1.3.4. DBConnection

Clase que provee la funcionalidad necesaria para que el subsistema de inferencia realice las conexiones a las bases de datos del corpus de usuarios y de la base de conocimiento. A continuación se expone una explicación detallada de los métodos de la clase.

- DBConnection(String, String, String, String, String,): Constructor de clase que almacena la ip, el usuario, la contraseña, el puerto y la instancia sobre la que se realizará la conexión.
- getCon(): Devuelve el objeto de tipo Connection con el que se enlaza con la base de datos.
- conectar(): Crea la conexión en función de los parámetros asignados en el constructor de clase.
- desconectar(): Realiza la desconexión de la base de datos.
- getTable(int): Método que nos devuelve el nombre de la tabla en la que se encuentra el usuario que recibe dentro del corpus de usuarios.
- readUser(int): Lee un usuario del corpus de usuarios.
- executeQuery(String): Ejecuta una consulta sobre la base de datos.
- executeInsert(String): Ejecuta una inserción sobre la base de datos.

6.1.3.5. UsersStore

Clase encargada del almacenamiento de los usuarios con los que se realiza la evaluación de los modelos. Almacenan el identificador del usuario actual, así como un Vector con el resto de usuarios que participan en la evaluación. Los métodos de la clase UsersStore son los siguientes:

- UsersStore(int, Vector<UserFeature>): Constructor de clase que recibe el usuario actual y el conjunto de usuarios participantes en la evaluación.
- getId_User(): Devuelve el identificador del usuario actual.
- setIdUser(int): Asigna un identificador al usuario actual.
- getStore(): Devuelve el almacén de usuario que participan en la evaluación.
- setStore(Vector<UserFeature>): Asigna un nuevo conjunto de usuarios al almacén.

- getFeature(int): Devuelve la característica a la que apunta el parámetro de tipo entero que recibe para el usuario actual.
- setFeature(UserFeature): Asigna un nuevo usuario con toda su información al almacén de usuarios.

6.1.3.6. InferrUsers

La última clase del subsistema de inferencia, "UserFeature" no se analizará, ya que su funcionamiento es análogo al explicado en el apartado 6.1.1.1.2 de la presente documentación.

6.1.4. Implementación subsistema de estructura

Tal y como se vio en el capítulo de diseño, el subsistema de estructura, se encarga de los servicios propios del modelo de usuario, así como de la estructura de éste. Los procedimientos básicos para ofrecer estos servicios son, start, drop, reset, además el modelo debe ser capaz aprender el contexto de un usuario y de facilitar información de un usuario. Para ello se emplean los procedimientos de abrir sesión que permite la entrada de un nuevo usuario en el modelo, el procedimiento cerrar sesión se encarga de aprender el contexto de un usuario y por último el procedimiento de pedir información de un usuario al modelo que realiza inferencia sobre el modelo. A continuación se exponen los diagramas de flujo de datos que modelan el comportamiento de la información en el modelo.

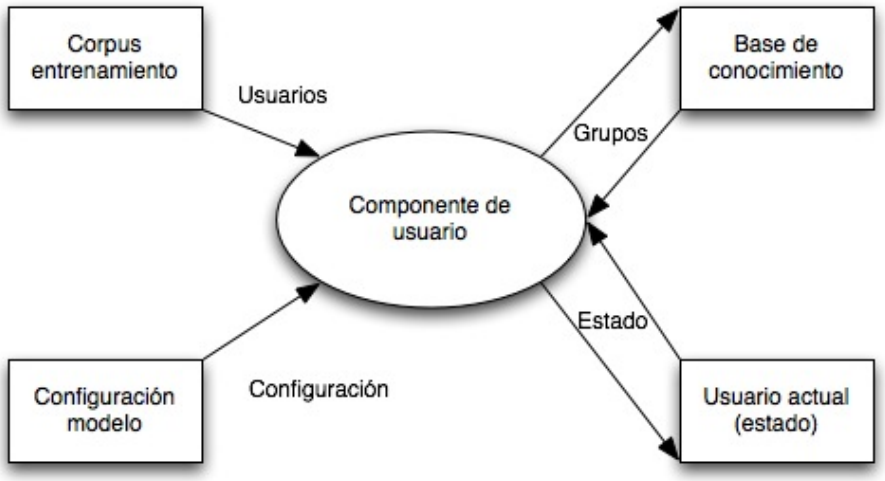


Ilustración 20. Diagrama de contexto

El diagrama anterior consiste en un diagrama de flujo de datos de nivel cero, es decir un diagrama de contexto en el que existe una entidad “Componente de usuario” y cuatro almacenes de datos. A continuación se detallan cada uno de los elementos del diagrama. La entidad “Componente de usuario”, permite representar los flujos de información entre el componente y los almacenes. En lo que se refiere a los almacenes, existen cuatro almacenes que se detallan seguidamente.

- Base de conocimiento: representa la base de conocimiento del modelo y almacena los grupos de usuario.
- Corpus entrenamiento: almacena los datos de los usuarios con los que se entrena el modelo.
- Configuración modelo: permite guardar la parametrización del modelo.
- Usuario actual: representa el estado del modelo en un momento dado.

Tras el análisis del diagrama de contexto, se expone el diagrama de flujo de datos de primer nivel con el fin de profundizar en el manejo de la información del sistema.

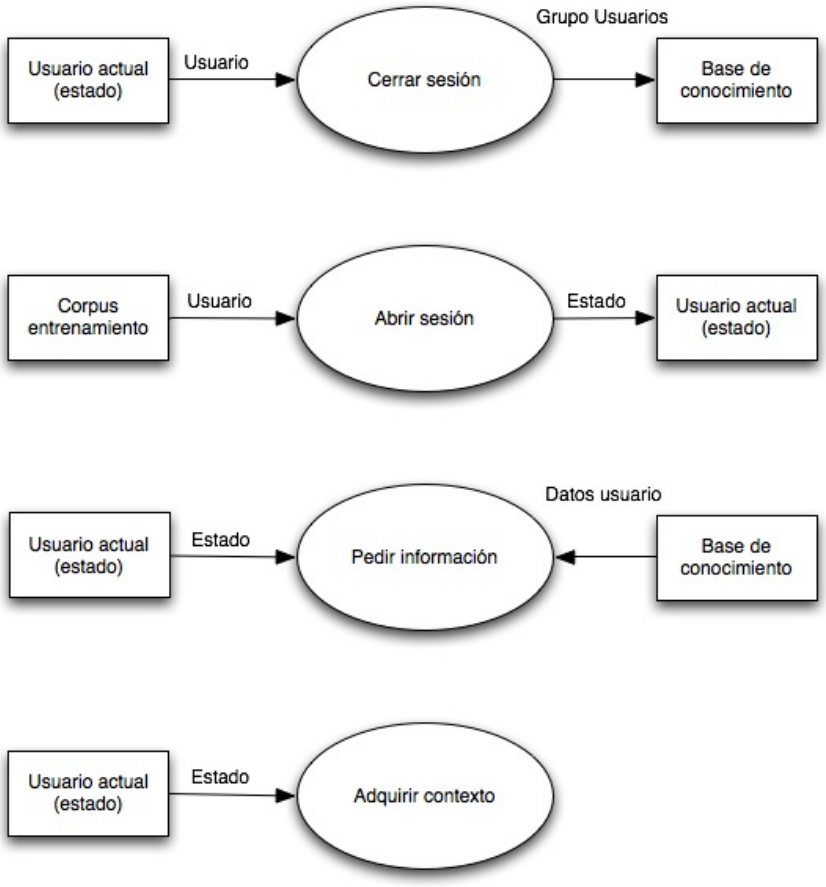


Ilustración 21. Diagrama de flujo de datos

A continuación se explican cada uno de los procesos mostrados en el diagrama de flujo de datos anterior.

- Cerrar sesión: permite al modelo aprender el contexto del usuario actual que queda almacenado en la base de conocimiento y a su vez cierra la sesión del usuario.
- Abrir sesión: permite a un nuevo usuario interaccionar con el modelo.
- Pedir información: posibilita la obtención de información sobre un usuario en un contexto.
- Adquirir contexto: el modelo aprende el contexto de un usuario, se realiza al cerrar sesión de un usuario y se puede realizar mediante la interacción al pedir información de un usuario al modelo.

6.1.5. Implementación subsistema cliente

Para el subsistema cliente se empleo un software de terceros, la descripción detallada de dicho software puede encontrarse en el documento “Desarrollo de infraestructuras para el modelado de usuarios” [del Coso, A. 2009].

6.2. Aspectos de necesidades de información

En lo que se refiere a implementación sobre las necesidades de información del modelo, esta parte se implemento usando Pl/Sql sobre el sistema gestor de bases de datos Oracle 11G. Si bien los script con la implementación Pl/Sql no se incluyen en esta memoria debido a su extensión y complejidad, si que se van a mencionar ciertas decisiones de implementación que se tomaron en cuanto al manejo de información en el modelo. Como ya se comento anteriormente, el número de grupos de usuarios máximo del modelo se fijó a cincuenta por cuestiones de rendimiento, de la misma forma se decidió fijar el número máximo de filas que posee la tabla que almacena los grupos de usuarios a ocho mil, así pues si en algún momento se superasen las ocho mil filas en la tabla bien por una inserción o bien por una fusión, se borrarían filas hasta llegar a la cota. El criterio que se sigue para eliminar filas es en función de la certeza, es decir, se eliminan las filas de menor certeza hasta que queden ocho mil, de esta forma se están eliminando las filas menos representativas y que por tanto menos información aportan al modelo. A continuación se expone, por tratarse de un fragmento aislado y particular, el código empleado para la eliminación de filas.

```
select count('x')-8000 into elim from tabla_raiz;

delete from tabla_raiz where (id_user, caracteristica, valor ) in
```

(select id_user, caracteristica, valor from (select abs(certeza) num, id_user,

caracteristica, valor from tabla_raiz order by (num) asc)

where rownum <= elim);

Como puede observarse el fragmento de código anterior, almacena en la variable de tipo number "elim" el número de filas menos ocho mil que existen en la tabla "tabla_raiz" que será la encargada de almacenar la base de conocimiento del modelo. Posteriormente se eliminan de dicha tabla todas aquellas filas cuyo "rownum" u ordenación física sea inferior a ocho mil, habiendo previamente ordenado por los valores de certeza en orden ascendente. Por lo tanto, se limita el número de filas de la tabla, lo que impide la existencia de grupos muy grandes que puedan dar lugar a fusiones muy complejas y que perjudicarían gravemente al rendimiento del modelo. Sin embargo al eliminar filas de la tabla hasta quedarse con las ocho mil más representativas se podría estar eliminando uno o varios grupos al completo, en este caso el modelo volvería a insertar el siguiente grupo en la posición del grupo eliminado con lo que la tabla volvería a tener en todo caso cincuenta grupos de usuarios.

6.3. Aspectos físicos y de comunicaciones

Este capítulo recoge los aspectos de físicos y de comunicaciones, es decir en él se expone la distribución física de los módulos del sistema, así como las comunicaciones que realiza el sistema. En cuanto a las comunicaciones que realiza el sistema para comunicar las distintas maquinas entre las que se distribuye, éstas se realizan mediante secure socket ya que las tecnologías empleadas (Java y Oracle) realizan conexiones seguras mediante sockets. A continuación se expone la distribución física definitiva del sistema.

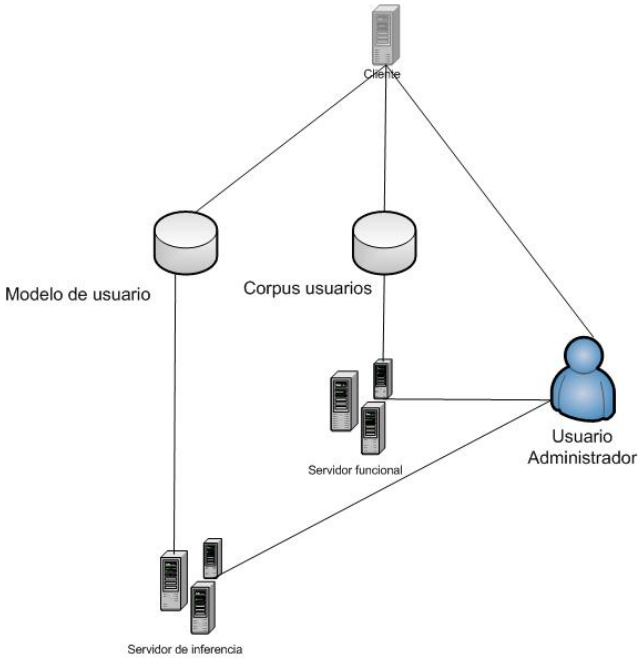


Ilustración 22. Diseño físico implementado

7. Validación y Evaluación

En la presente sección, se desarrollará la validación y la evaluación llevada a cabo sobre los dos modelos propuestos a lo largo del documento. Por tratarse de un modelo de usuario auto modelado, previo a la validación y a la evaluación de éste es necesario realizar un entrenamiento de ambos modelos, por lo tanto esta sección también ofrece al lector una visión detallada de ello. Seguidamente se detallaran los procesos de validación y evaluación del sistema.

7.1. Dominio y datos

En primer lugar se analizarán los datos y el dominio sobre el cual se realizará el entrenamiento del modelo. Los datos de entrenamiento, así como los del proceso de evaluación, son datos reales extraídos de una red social mediante un web crawler, sin embargo, dichos datos son de gran diversidad por lo que podrían dificultar la evaluación del modelo. A continuación se exponen las principales características de este conjunto de datos.

Número de usuarios	500.000
Número de características	23
Número de valores máximo para una característica	177.851
Número de valores mínimo para una característica	57
Número medio de valores para una característica	41.788
Número total de valores de las características	961138

Tabla 2. Características del dominio

Tal y como puede observarse en la tabla anterior, la disparidad de los datos con los que se va a trabajar es muy grande ya que el número de valores por característica prácticamente dobla el número de usuarios existentes. Por lo tanto, y teniendo en cuenta la diversidad del domino, se espera que el conjunto de datos de trabajo cuente con un número elevado de valores atípicos que afectarán notablemente el resultado de los experimentos.

7.2. Entrenamientos

El entrenamiento realizado fue el mismo para los dos modelos planteados, el entrenamiento se realizó sobre cien mil usuarios elegidos del total de usuarios extraídos de la red social. Así pues de un total de quinientos mil usuarios, se va a realizar el entrenamiento sobre un subconjunto de cien mil, esta decisión se tomo en base al volumen de datos de entrenamiento, así como a los tiempos de entrenamiento.

En primer lugar destacar que una muestra de cien mil usuarios, es una muestra suficientemente representativa como para poder valorar el funcionamiento del modelo. En segundo lugar, los tiempos esperados de entrenamiento del modelo son altos y crecen a medida que aumenta la información del modelo, por lo tanto se considera que una muestra de cien mil usuarios es suficientemente representativa para validar el modelo y que el tiempo que el modelo tarda en aprender cien mil usuarios, no es excesivo y permite obtener resultados en un tiempo razonable. De esta manera, tras este entrenamiento se procedió a la validación y evaluación del modelo.

7.3. Validación

La validación del modelo se desarrolla en dos fases, una primera fase durante el entrenamiento del modelo, en la que se comprueba que el modelo es capaz de "automodelarse", es decir, capaz de crear estereotipos para este dominio concreto. En la segunda fase tras el entrenamiento del modelo, se comprueba que éste es capaz de inferir información para un nuevo usuario en base al conocimiento que el modelo tiene de este usuario, así como en base al estereotipo que mejor se ajusta al nuevo usuario, es decir el grupo en el que mejor encaja el usuario. Tras la validación del sistema se pasa a la tarea de evaluación donde se comprobara como de bueno es el modelo en un dominio concreto.

7.4. Entorno de experimentación

En esta sección se pretende describir y justificar el entorno elegido para la experimentación que se va llevar a cabo con los modelos de usuario implementados. Tal y como puede verse en la ilustración 22, los distintos módulos del sistema se distribuyeron en distintas maquinas con el fin de mejorar la eficiencia del sistema. A continuación se expone una descripción de las máquinas empleadas, así como el componente del modelo de usuario que albergan cada una de ellas.

Componente	Sistema operativo	Arquitectura	SGBD	Memoria	Procesador
Subsistema de inferencia	Microsoft Windows XP	32 bits		4GB	Intel core2 duo 2.4GHz
Corpus usuarios	Microsoft Windows XP	32 bits	Oracle 11G	2GB	AMD Athlom 2.7GHz
Subsistema cliente	Microsoft Windows XP	32bits		2GB	Intel Pentium4 3GHz
Componente de usuario	Windows server 2008	64 bits	Oracle 11G	8GB	Intel Xeon E5450 3GHz
Base de conocimiento	Windows server 2008	64 bits	Oracle 11G	8GB	Intel Xeon E5450 3GHz

Tabla 3. Distribución física

La distribución física anterior responde a cuestiones de eficiencia del sistema, como puede observarse, la máquina más potente de las empleadas será la responsable del componente de usuario que es el que mayor complejidad computacional requiere.

7.5. Parámetros experimentación

Para la evaluación de ambos modelos se plantearon cuatro experimentos por modelo, cada experimento es idéntico a su análogo en el otro modelo, con el fin de poder comparar los resultados de ambos. Las siguientes tablas muestran las características de cada experimento:

	Experimento 1	Experimento 2	Experimento 3	Experimento 4
Datos	Dentro de rango	Fuera de rango	Dentro de rango	Fuera de rango
Número usuarios	20000	20000	20000	20000
Iteraciones/usuario	18	18	18	18
Rango valores	[0,1]	[0,1]	[0,1]	[0,1]
Mejor coincidencia	Máximo encaje	Máxima certeza	Máxima certeza	Máximo encaje

Tabla 4. Experimentos modelo 1

	Experimento 5	Experimento 6	Experimento 7	Experimento 8
Datos	Dentro de rango	Fuera de rango	Dentro de rango	Fuera de rango
Número usuarios	20000	20000	20000	20000
Iteraciones/usuario	18	18	18	18
Rango valores	[-1,1]	[-1,1]	[-1,1]	[-1,1]
Mejor coincidencia	Máximo encaje	Máxima certeza	Máxima certeza	Máximo encaje

Tabla 5. Experimentos modelo 2

A continuación se explican los parámetros de cada experimento, con el fin de analizar las diferencias entre ambos experimentos y los resultados de estos.

- Datos: Los datos de los experimentos pueden ser, o bien un subconjunto de los datos con los que se realizo el entrenamiento, o bien un subconjunto del total de datos, y dicho subconjunto debe ser disjunto con el conjunto de datos del entrenamiento. Así pues, en la tabla anterior, en la fila "datos", dentro de rango se refiere a un subconjunto de los datos del entrenamiento, mientras que fuera de rango se refiere a un subconjunto de datos disjunto del conjunto de datos de entrenamiento.
- Número de usuario: Este parámetro indica los usuarios con los que se realiza el experimento y es el mismo para todos los experimentos.
- Iteraciones/usuario: Indica el número de iteraciones que se realizan para cada usuario, es decir el número de características que se aprenden para cada usuario.
- Mejor coincidencia: Este parámetro indica que nos quedaremos, o bien con el grupo de usuarios que mejor encaje con el actual, o bien con aquel que nos ofrezca una mayor certeza.

7.6. Dinámica de los experimentos

A lo largo de este apartado se ofrece una visión de cuál es la dinámica de los experimentos anteriormente planteados. Para el caso de los cuatro experimentos planteados, se trabaja sobre un subconjunto de veinte mil usuarios, así pues el primer paso de los experimentos será elegir un usuario aleatorio, para dicho usuario se fijará una característica que es sobre la que se desea realizar inferencia, destacar que la característica a fijar es también aleatoria. Tras realizar este proceso, se preguntará al modelo por esa característica de ese usuario, hay que

destacar también, que en este momento el modelo no tiene ningún conocimiento del usuario. La respuesta obtenida por el modelo se almacenará posteriormente el modelo aprende una característica del usuario, resaltar que en ningún caso la característica aprendida puede ser por la que se está preguntando, pues a continuación se vuelve a preguntar al modelo por la característica fijada y se obtiene otra respuesta que se almacena. El proceso se repite hasta que el modelo conoce un máximo de dieciocho características del usuario, con lo que es en esta situación cuando es esperable que los resultados del modelo sean óptimos, ya que es cuando el modelo conoce más información del usuario con el que interactúa.

Llegado este punto hemos de justificar las decisiones tomadas en la elección de los valores de los parámetros de la experimentación. En primer lugar el número de usuarios sobre el que se realiza la experimentación fue elegido por tratarse de una muestra representativa sobre el total de usuarios con los que se realizó el entrenamiento, que era de cien mil usuarios. A continuación pasamos a analizar el número de iteraciones de los experimentos que es de dieciocho por usuario. Se deben tener en cuenta los siguientes parámetros para entender esta decisión.

Máximo características por usuario	21
Media características por usuario	11
Mínimo de características por usuario	3

Tabla 6. Parámetros dominio

Teniendo en cuenta los datos de la tabla anterior, parecería lógico tomar un número de iteraciones próximo a once, sin embargo, esto penalizaría muchísimo a los usuarios con mayor numero de características, por este motivo se decidió realizar los experimentos con dieciocho iteraciones o características aprendidas por el modelo, ya que no se penaliza a usuarios con muchas características y ya que los usuarios con menos características, lo único que sucede es que no se puede aprender más información de ellos. Por lo tanto, la respuesta del modelo sería siempre la misma a partir de que se aprendió la última característica del usuario.

7.7. Experimentación

A continuación se exponen y se analizan los resultados de los distintos experimentos sobre cada uno de los modelos propuestos. A partir de ahora nos referiremos como modelo1 al modelo que procesa valores dentro del rango [0,1] y llamaremos modelo2 al modelo que realiza

el tratamiento de valores dentro del rango [-1,1]. Así pues, y en referencia a los experimentos propuestos anteriormente, los experimentos uno, dos tres y cuatro aplican sobre el modelo1 mientras que los experimentos cinco seis siete y ocho aplican sobre el modelo2.

7.7.1.Experimento 1

El experimento uno tal y como se definió anteriormente, aplica sobre el modelo1, a continuación se exponen todos los parámetros del experimento.

Nombre	Experimento1
Datos	Dentro de rango
Número usuarios	20000
Iteraciones/usuario	18
Mejor coincidencia	Máximo encaje
Modelo	Modelo1
Rango modelo	[0,1]

Tabla 7. Experimento 1

Tras los datos del primer experimento se expone la gráfica de resultados del proceso de evaluación del primer experimento.

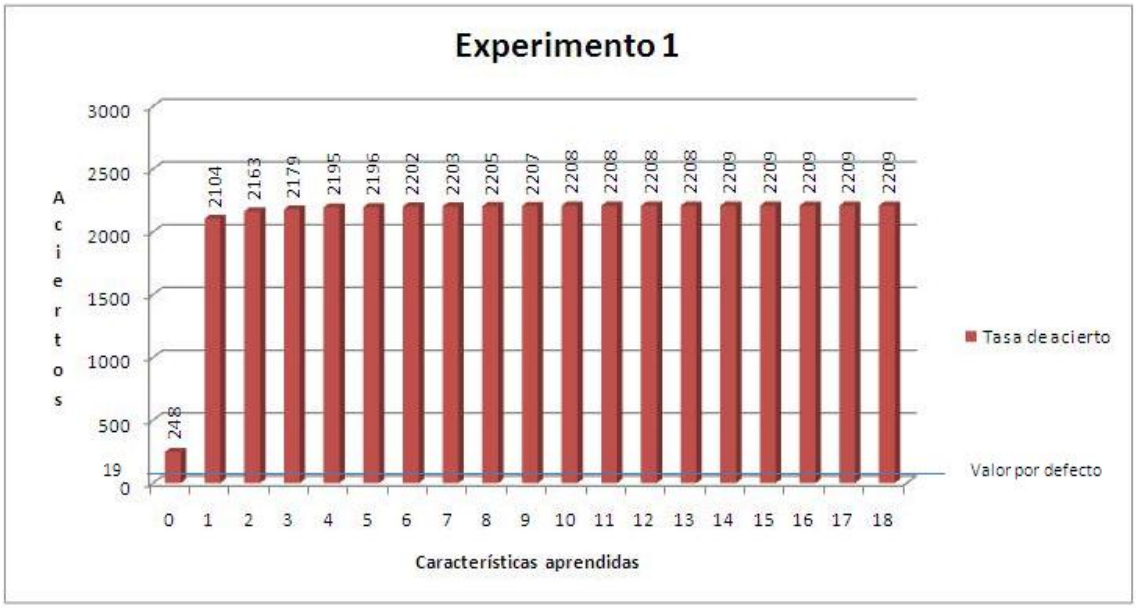


Ilustración 23. Resultados experimento 1

A continuación se analizarán los resultados obtenidos durante el primer experimento de la evaluación del modelo. Por tratarse del primer experimento y la primera gráfica de resultados expuesta en este documento, se comentarán a su vez ciertos datos inherentes a la propia gráfica con el fin de facilitar al lector la comprensión de la totalidad de las gráficas de evaluación. En primer lugar se van a especificar las medidas de cada uno de los ejes de la gráfica, el eje de abscisas o eje "x", esta numerado de cero a dieciocho, este patrón será constante en el resto de gráficas de evaluación, ya que este eje siempre representa el número de características que el modelo conoce del usuario. Puesto que todos los experimentos se planificaron hasta un máximo de dieciocho características aprendidas por el modelo, la numeración del eje será siempre la misma. En cuanto a los valores del eje de abscisas, solamente existe un valor a destacar que es el cero, en este caso se pregunta al modelo por una característica del usuario actual, sin embargo el modelo no sabe nada de dicho usuario, por lo que el modelo dará una respuesta por defecto en este caso. Según el experimento emplee la técnica de máximo encaje o máxima certeza, dicho valor por defecto será uno u otro.

En cuanto al eje de ordenadas, sus rangos se fijaron de cero a tres mil, en este eje se muestran los aciertos del modelo para cada iteración del experimento. Conviene en este punto recordar que la totalidad de experimentos propuestos se planificaron para veinte mil usuarios, por lo que la tasa de acierto máxima es de veinte mil. El eje de abscisas también muestra un valor atípico en todas las gráficas, se trata del valor diecinueve, este valor indica el número de usuarios medio que contienen el valor más frecuente de cada característica, lo que nos ofrece

una visión global de la diversidad del dominio. Como puede observarse el valor por defecto, es siempre inferior a la tasa de acierto mínima del modelo. Dado que el valor por defecto se sitúa en diecinueve aciertos sobre veinte mil, podemos concluir que el dominio de los datos es muy abierto, lo que dificulta el buen funcionamiento del modelo por estar en un dominio con mucha diversidad.

En lo que se refiere al primer experimento, puede observarse en la gráfica que el mayor aprendizaje del modelo se realiza en la primera iteración, es decir en el paso del desconocimiento al conocimiento de una característica del usuario. La tasa tan baja en el caso de desconocimiento, es decir en la primera iteración, se debe a la diversidad del dominio, así como a la técnica empleada que recordemos que para este caso es quedarse con el usuario de mayor encaje, de esta manera el modelo en este caso devolverá el grupo por defecto, sin tener en cuenta la característica por la que se pregunta, ya que el modelo no conoce nada del usuario. Como podrá observarse más adelante, los experimentos que empleen la técnica de mayor certeza no producirán un resultado tan bajo en la primera iteración ya que al considerarse el usuario de mayor certeza, si que se considera la característica por la que se pregunta y se devuelve el valor por defecto para esa característica, que siempre es mejor que devolver el grupo por defecto sin tener en cuenta la característica tal y como hacen los experimentos de mayor encaje.

También conviene destacar que al aprender una característica de los usuarios, el modelo es capaz de responder adecuadamente en dos mil catorce de los veinte mil casos, es decir, que con tan solo una característica aprendida de los usuarios el modelo tiene una tasa de acierto superior al 10%, concretamente del 10,7%. A su vez, si analizamos el resto de la gráfica, se puede observar que para el resto de características no existen grandes mejoras, pues si bien en las nueve primeras características aprendidas se puede observar una mejora en todas ellas, sin embargo, a partir de la novena iteración el modelo se estanca, experimentando un pequeña mejora en la decimotercera iteración que le lleva a alcanzar el máximo del experimento situado en un 11%.

Por lo tanto, las conclusiones que se obtienen en este primer experimento, son que el modelo puede ofrecer una baja tasa de acierto en el caso de desconocimiento, especialmente esto puede ocurrir si se emplea la técnica de mayor encaje para seleccionar a los usuarios. Esta baja tasa de aciertos se debe a la complejidad y diversidad del dominio tal y como se explicó con anterioridad.

7.7.2.Experimento 2

El experimento dos se realiza sobre el modelo1, la siguiente tabla muestra los datos del experimento.

Nombre	Experimento2
Datos	Fuera de rango
Número usuarios	20000
Iteraciones/usuario	18
Mejor coincidencia	Máxima certeza
Modelo	Modelo1
Rango modelo	[0,1]

Tabla 8. Experimento 2

A continuación se expone la gráfica de resultados del segundo experimento, este experimento se realizó sobre el mismo modelo que el experimento 1, se empleó la técnica de mayor certeza y se emplearon datos distintos a los que se usaron en el entrenamiento del modelo.

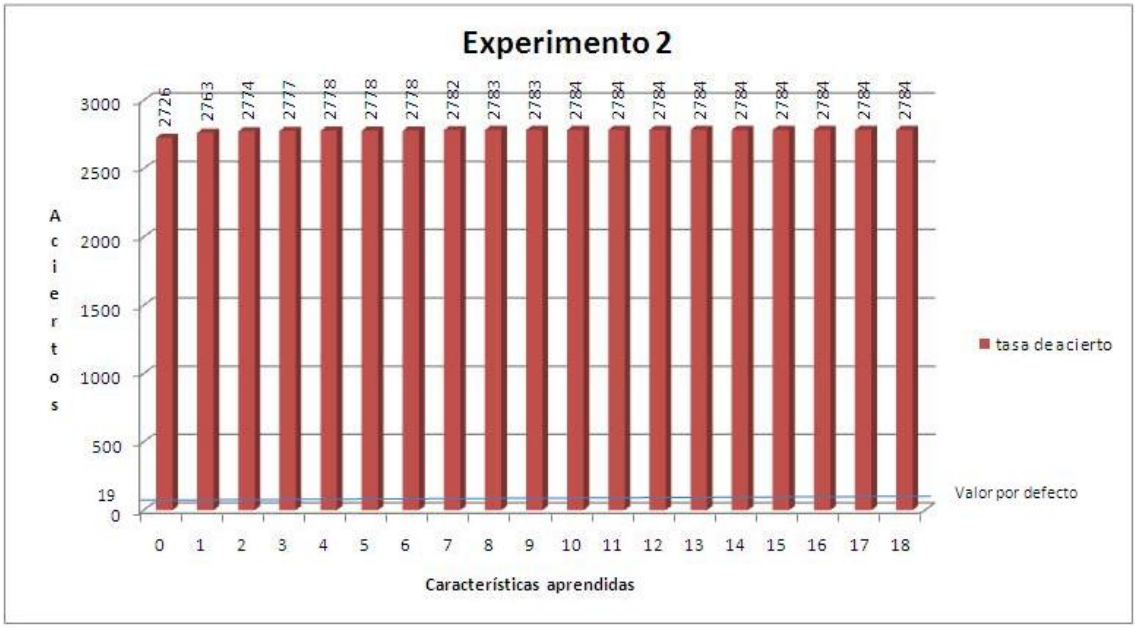


Ilustración 24. Resultados experimento 2

Como puede observarse en la gráfica anterior, los resultados del experimento son prácticamente lineales, además debe observarse que aun tratándose de datos distintos a los del entrenamiento de modelo, las tasas de aciertos son mayores que las del experimento 1, esto es debido a varias razones, en primer lugar en este segundo experimento se emplea la técnica de máxima certeza. Por otra parte, tal y como se analizo en el apartado anterior del presente documento, el dominio de los datos es muy diverso, lo que puede posibilitar que los experimentos fuera de rango proporcionen resultados iguales o mejores que los dados por los experimentos realizados dentro de rango. Se puede afirmar que si los experimentos se realizasen sobre una muestra de datos mayor de veinte mil usuarios, los resultados de los experimentos dentro y fuera de rango tenderían a ser más parecidos a medida que se amplía la muestra.

Las conclusiones que se obtienen de este segundo experimento son que a diferencia del experimento anterior no se obtiene un resultado muy bajo en el caso de desconocimiento, esto es debido a la técnica empleada que en este caso es la de mayor certeza, esta técnica, como ya se explico, devolverá el valor por defecto teniendo en cuenta la característica por la que se pregunta, por lo que el resultado de la primera iteración es mayor que en los casos de mayor encaje.

También se puede concluir que de la misma forma que en el experimento anterior, el mayor aprendizaje se produce en la primera iteración, si bien en este caso la mejoría

experimentada es mucho más suave que en el caso anterior. En cuanto a las tasas de acierto, estas son en todo caso superiores al 13% alcanzando un máximo de 13,92% en la iteración diez.

7.7.3.Experimento 3

El experimento tres se realiza sobre el modelo1, a continuación se exponen todos los parámetros del experimento.

Nombre	Experimento3
Datos	Dentro de rango
Número usuarios	20000
Iteraciones/usuario	18
Mejor coincidencia	Máxima certeza
Modelo	Modelo1
Rango modelo	[0,1]

Tabla 9. Experimento 3

Tras exponer las principales variables que intervienen en el experimento, se expone la gráfica de resultados del experimento.

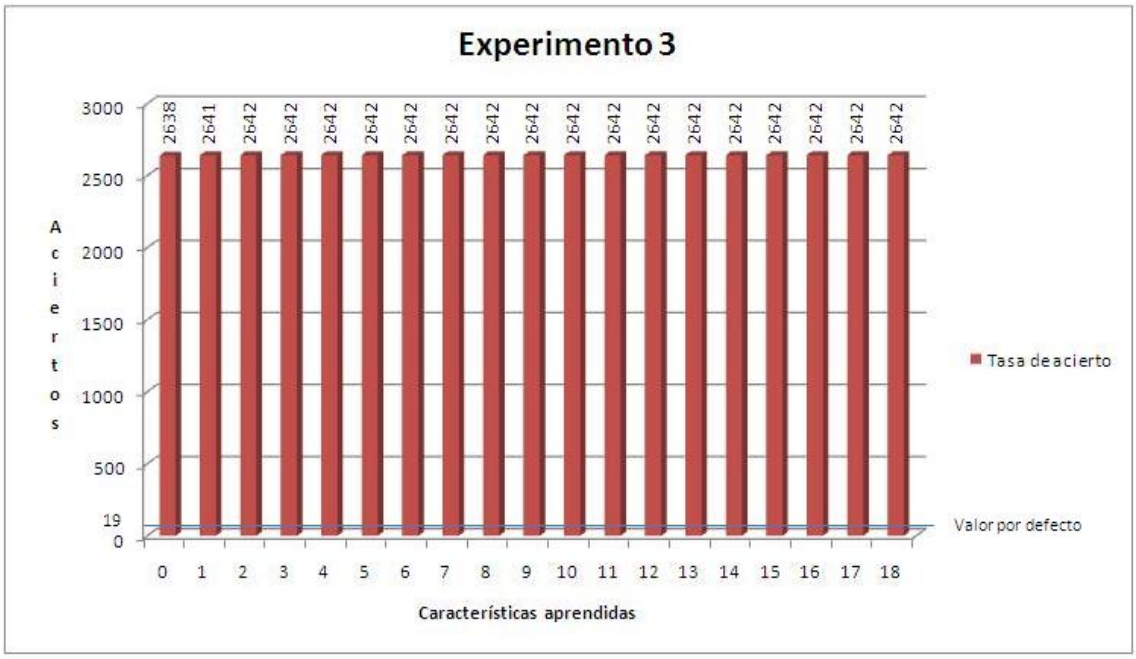


Ilustración 25. Resultados experimento 3

Tal y como puede observarse, los resultados del experimento son ligeramente inferiores a los obtenidos en el experimento anterior que se realizo sobre el mismo modelo y empleando la misma técnica, solo que con datos fuera del rango de entrenamiento, como ya se justificó anteriormente, esta situación se da por la diversidad del dominio de los datos. La tasa máxima de acierto del experimento es del 13,21%, que si bien es más baja que la del experimento anterior, es una tasa relativamente buena dado el dominio en que se trabaja. La tasa máxima de acierto se produce en la iteración décima, de la misma forma que sucedía en el experimento anterior, con lo que se puede afirmar que a partir de la décima característica aprendida por el modelo, el aprendizaje se estanca. De la misma forma que sucedía con gráficas anteriores en las que se empleo la técnica de máxima certeza la primera iteración o desconocimiento, ofrece resultados muy similares a los dados por el resto de iteraciones.

7.7.4. Experimento 4

El experimento cuatro aplica sobre el modelo1, la siguiente tabla muestra los principales datos del experimento.

Nombre	Experimento4
Datos	Fuera de rango
Número usuarios	20000
Iteraciones/usuario	18
Mejor coincidencia	Máximo encaje
Modelo	Modelo1
Rango modelo	[0,1]

Tabla 10. Experimento 4

Tras la tabla de parámetros del experimento, se muestra la gráfica de resultados de este.

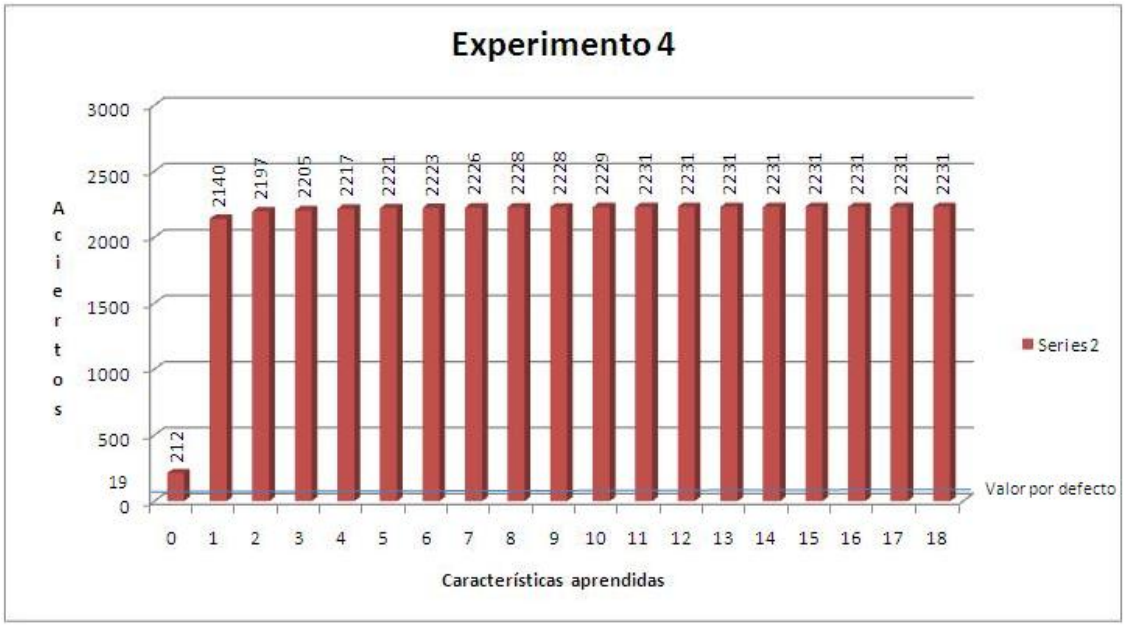


Ilustración 26. Resultados experimento 4

El presente experimento es similar al experimento uno, pues emplea la misma técnica sobre el mismo modelo, con la única diferencia de que en este caso se utilizan datos fuera de rango. Como ya se vio en experimentos anteriores, los resultados del experimento con datos fuera de rango son ligeramente superiores a los obtenidos en su análogo con datos dentro de rango. Puesto que este resultado ya se justifico anteriormente atendiendo a razones inherentes al

dominio de los datos, se procede a comentar las principales conclusiones que se pueden extraer del experimento.

Destacar que de la misma forma que el experimento uno, se obtiene una tasa de acierto baja para la iteración de desconocimiento, esto de nuevo es debido a la técnica empleada de mayor encaje, que para el caso de desconocimiento no es capaz de ofrecer resultados adecuados. Si bien el grupo por defecto que devuelve esta técnica, que será el mismo grupo para todos los experimentos en caso de desconocimiento, ofrece valores superiores al valor por defecto del dominio, que sitúa su tasa de acierto en diecinueve usuarios.

7.7.5.Experimento 5

El experimento cinco aplica sobre el modelo2, la siguiente tabla muestra los principales datos del experimento.

Nombre	Experimento5
Datos	Dentro de rango
Número usuarios	20000
Iteraciones/usuario	18
Mejor coincidencia	Máximo encaje
Modelo	Modelo2
Rango modelo	[-1,1]

Tabla 11. Experimento 5

Tras exponer los parámetros del experimento cinco, que es el primero que se realizó sobre el modelo2 que recordemos realiza un tratamiento de datos en el rango [-1,1], se exponen los resultados del experimento.

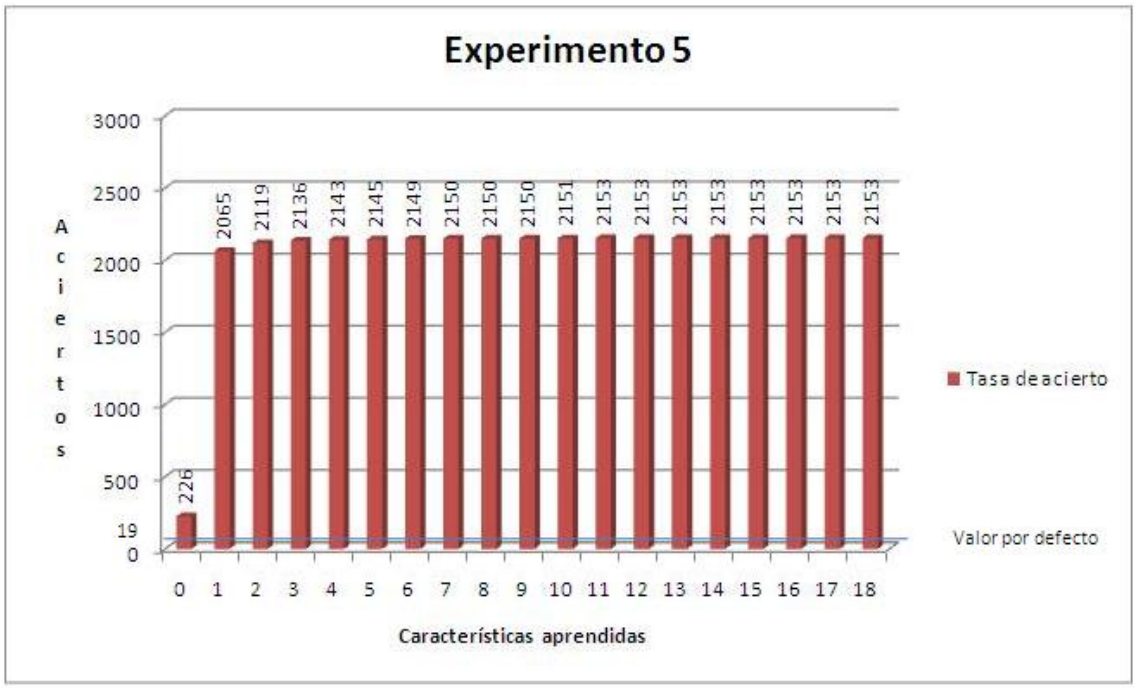


Ilustración 27. Resultados experimento 5

En cuanto al experimento cinco destacar que su análogo sobre el modelo uno, es el experimento uno. El experimento cinco tiene exactamente las mismas características que el experimento uno, solo que el modelo sobre el que se llevó a cabo realiza un tratamiento de los datos diferente al del experimento uno. En cuanto a los resultados que muestra la gráfica, cabe destacar que estos son ligeramente inferiores a los del experimento uno, por lo que en este caso se puede concluir que el modelo cuyo rango de valores es [0,1] funciona mejor que el tratamiento de valores en el rango [-1,1].

7.7.6. Experimento 6

El sexto experimento realizado fue sobre el modelo dos y a continuación se muestran sus principales parámetros.

Nombre	Experimento6
Datos	Fuera de rango
Número usuarios	20000
Iteraciones/usuario	18
Mejor coincidencia	Máxima certeza
Modelo	Modelo2
Rango modelo	[-1,1]

Tabla 12. Experimento 6

La siguiente gráfica muestra los resultados del sexto experimento.

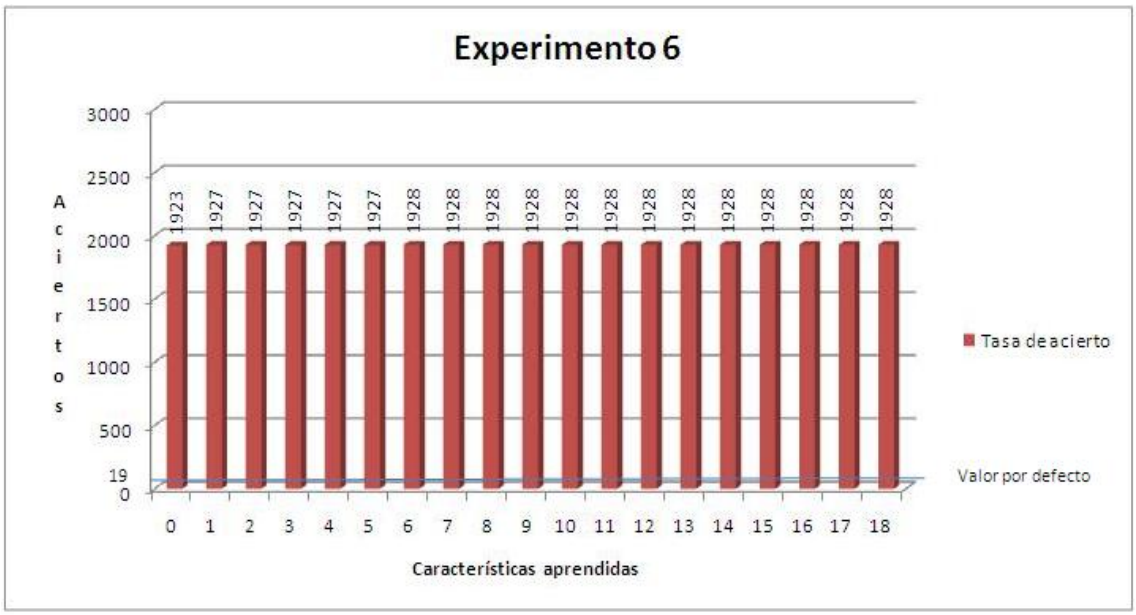


Ilustración 28. Resultados experimento 6

Tal y como puede observarse en la gráfica y atendiendo a los resultados del segundo experimento o experimento análogo, se puede observar que de nuevo los valores obtenidos para el rango de valores [-1,1] son inferiores a los obtenidos en el rango [0,1]. Tal y como se analizó en experimentos anteriores, se sigue produciendo la misma tendencia para el caso de desconocimiento, en el que los experimentos que emplean la técnica de mayor encaje y devuelven por tanto en este caso el grupo por defecto, reportan valores muy bajos de acierto para el desconocimiento, mientras que los experimentos que emplean la técnica de mayor certeza, siguen reportando valores elevados.

7.7.7.Experimento 7

El experimento siete aplica sobre el modelo2, la siguiente tabla muestra los principales datos del experimento.

Nombre	Experimento7
Datos	Dentro de rango
Número usuarios	20000
Iteraciones/usuario	18
Mejor coincidencia	Máxima certeza
Modelo	Modelo2
Rango modelo	[-1,1]

Tabla 13. Experimento 7

La siguiente gráfica muestra los resultados de la experimentación realizada según los parámetros anteriormente mostrados.

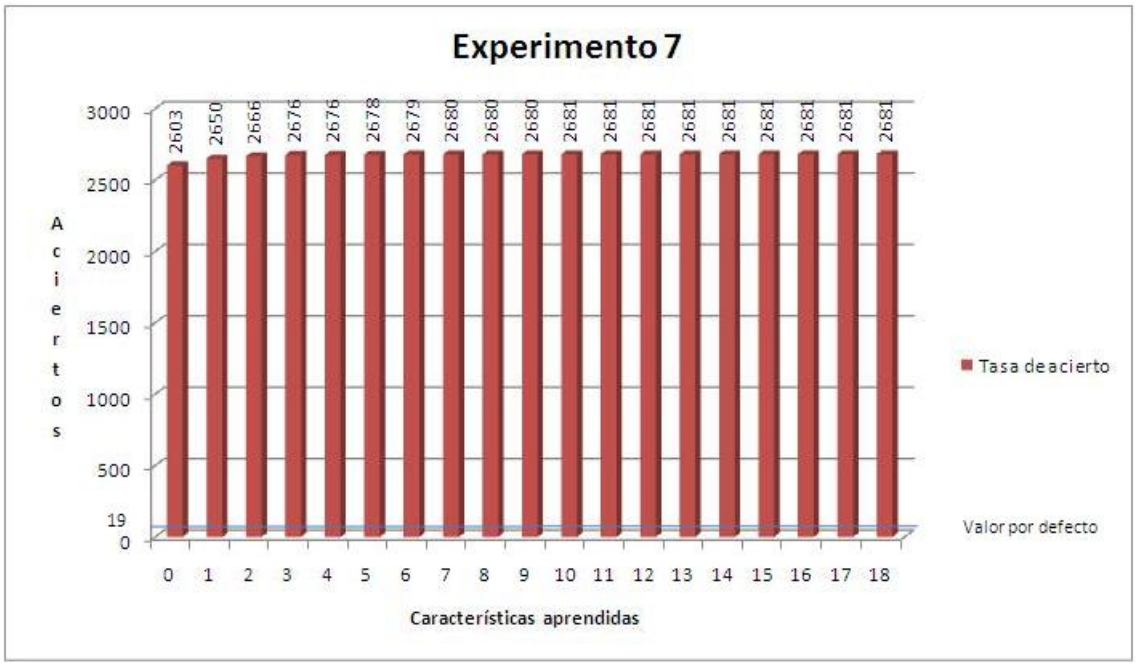


Ilustración 29. Resultados experimento 7

En el caso del experimento siete, no se observa una gran diferencia en las tasas de acierto con respecto a las obtenidas en el experimento tres, sino que en este caso ambos modelos se comportan de forma similar, para entrenamientos análogos.

7.7.8.Experimento 8

El experimento ocho aplica sobre el modelo2, la siguiente tabla muestra los datos.

Nombre	Experimento8
Datos	Fuera de rango
Número usuarios	20000
Iteraciones/usuario	18
Mejor coincidencia	Máximo encaje
Modelo	Modelo2
Rango modelo	[-1,1]

Tabla 14. Experimento 8

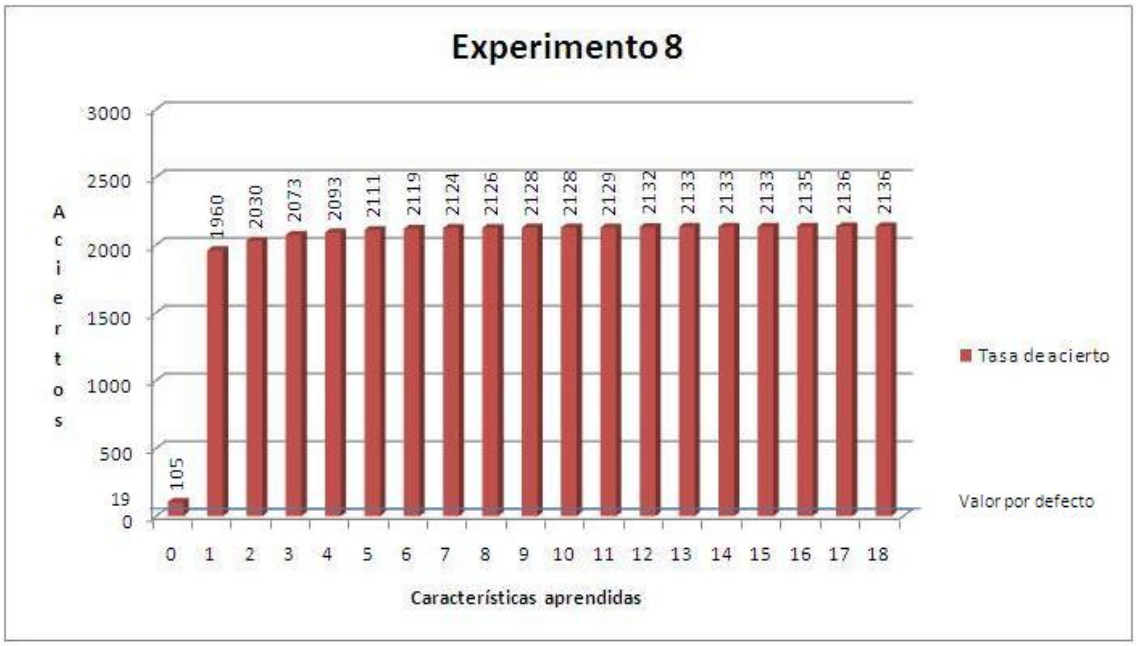


Ilustración 30. Resultados experimento 8

De forma similar a como sucedía con el experimento anterior, entre el experimento ocho y su análogo sobre el modelo uno, es decir, el experimento cuatro, no existen grandes diferencias en cuanto a las tasas de acierto.

7.7.9. Análisis de resultados y experimentos posteriores

Tras la experimentación anterior, se pueden extraer varias conclusiones que podrían llegar a ser contradictorias, en primer lugar destacar que si comparamos los experimentos mostrados en las tablas cuatro y cinco del presente documento, en tres de las cuatro comparaciones se comporta mejor el modelo en rango [0,1] que el modelo en rango [-1,1]. Sin embrago, si atendemos a las características de ambos modelos, intuitivamente el modelo en el rango [-1,1] debería funcionar mejor que el modelo en rango [0,1].

Como se analizó en los capítulos 5.1.1 y 5.1.2 del presente documento, las formulas de ambos dominios son equivalentes y congruentes, es decir, se trata de una conversión matemática de las formulas de un dominio al dominio contrario, lo que reduce las diferencias en el funcionamiento de los modelos a los redondeos. Intuitivamente, los redondeos deberían ser siempre favorables al modelo en [-1,1] ya que como se analizará a continuación este modelo tendrá un bit más de representación que el modelo en [0,1].

Si recordamos las tecnologías empleadas para la implementación del modelo de usuario, eran Java y Oracle, a su vez se conoce que el estándar de representación de ambas tecnologías

para operaciones en coma flotante es el estándar IEEE 754, es decir, coma flotante de precisión doble o sesenta y cuatro bits. Por lo tanto, ambas tecnologías representan los números en coma flotante empleando un bit de signo que será siempre usado, once bits de exponente y 52 bits de mantisa. Si tenemos en cuenta que el bit de signo en el caso del rango [0,1] siempre adoptara su representación positiva, se tiene que es un bit inservible en este modelo, en otras palabras un bit sin información, mientras que el modelo en [-1,1] siempre aprovecha todos sus bits por lo que su capacidad de representación es mayor. Así pues teniendo en cuenta esto y que la única diferencia entre ambos modelos son los redondeos, estos deberían favorecer al modelo con mayor capacidad de representación, sin embargo, la experimentación realizada nos demuestra que no es así. Tras el análisis anterior se decidió analizar los experimentos propuestos con el fin de explicar porqué el modelo en [0,1] se comporta mejor que el modelo en [-1,1] cuando intuitivamente debería ser al revés.

Teniendo en cuenta los datos anteriormente expuestos, se reduce la aleatoriedad de los experimentos realizados a la selección de los veinte mil usuarios con los que se entrenará, así como la característica que se fija para cada uno de ellos, sin embargo, a priori se considera que la muestra de veinte mil usuarios sobre cien mil debería ser suficientemente representativa como para que la aleatoriedad no influyese en los resultados, aun así se decidió plantear dos nuevos experimentos en los que se tomarían los mismos usuarios para el modelo [0,1] que para el modelo [-1,1], con el fin de determinar si la muestra tomada aleatoriamente influye o no en los resultados. A continuación se muestran los datos de los dos nuevos experimentos.

7.7.9.1. Experimento 9

El experimento nueve aplica sobre el modelo2, la siguiente tabla muestra los principales datos del experimento.

Nombre	Experimento9
Datos	Dentro de rango
Número usuarios	20000
Iteraciones/usuario	18
Mejor coincidencia	Máximo encaje
Modelo	Modelo2
Rango modelo	[-1,1]

Tabla 15. Experimento 9

Tras el análisis de los parámetros del experimento nueve, que se realizó sobre el modelo en [-1,1], se expone la gráfica con los resultados obtenidos en la experimentación.

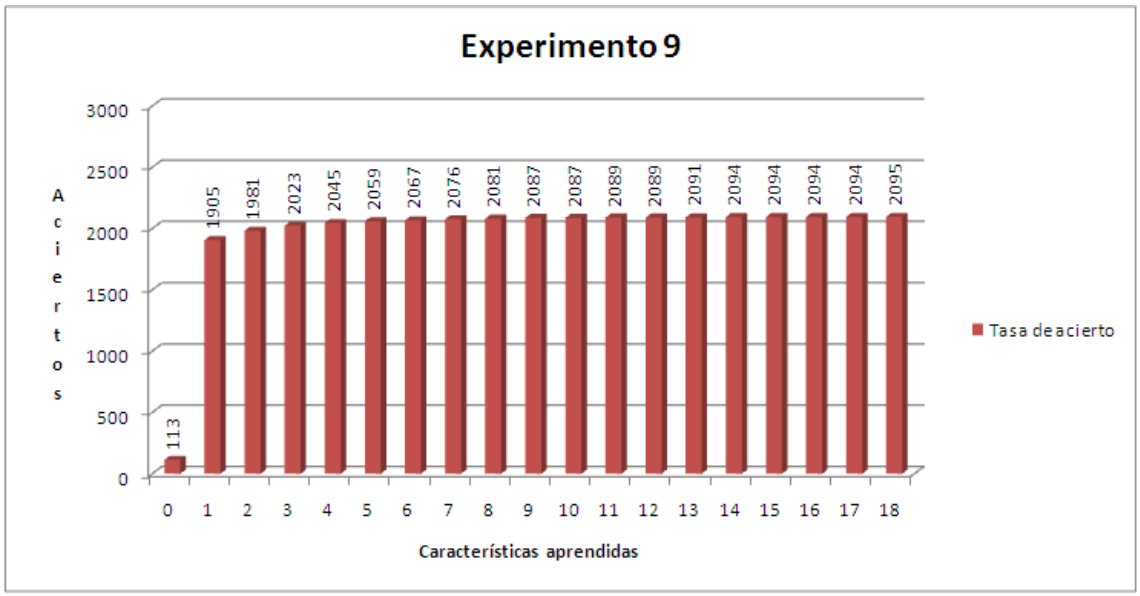


Ilustración 31. Resultados experimento 9

Como puede observarse los resultados obtenidos son bastante similares a los obtenidos en otros experimentos realizados sobre este modelo, si bien lo que nos interesa es comparar los resultados de este experimento con los del experimento diez, cuyos datos se muestran a continuación, se recuerda al lector que el experimento diez es exactamente igual al experimento nueve, en lo que a muestra de usuarios se refiere, y en cuanto a características fijadas para cada usuario.

7.7.9.2. Experimento 10

El experimento diez aplica sobre el modelo1, la siguiente tabla muestra los principales datos del experimento.

Nombre	Experimento10
Datos	Dentro de rango
Número usuarios	20000
Iteraciones/usuario	18
Mejor coincidencia	Máximo encaje
Modelo	Modelo1
Rango modelo	[0,1]

Tabla 16. Experimento 10

A continuación mostramos la gráfica de resultados del experimento diez.

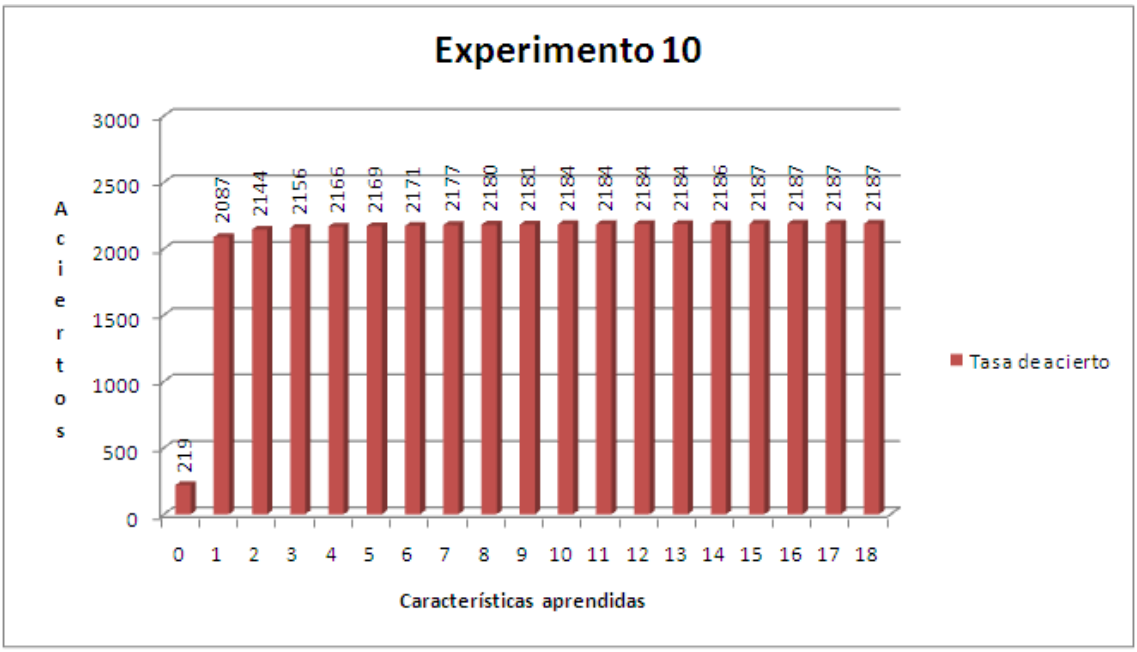


Ilustración 32. Resultados experimento 10

Tal y como puede observarse comparando los resultados de la ilustración treinta y uno con los resultados obtenidos en el experimento diez, el modelo en [0,1] vuelve a ser mejor que el modelo en [-1,1], lo que nos permite justificar que la muestra de veinte mil usuarios elegida es suficientemente representativa y la aleatoriedad con que se eligen dichos usuarios no está influyendo en los resultados de los experimentos. Así pues, y tras comprobar que la muestra seleccionada no influye en los resultados de la experimentación, los esfuerzos se centraran en analizar la distribución de ambos modelos tras el entrenamiento con el fin de explicar porqué el modelo en [0,1] es mejor que el modelo en [-1,1], cuando intuitivamente debería ser al revés.

7.7.9.3. Análisis distribución modelos

A continuación se realiza un análisis exhaustivo de los modelos entrenados con el fin de explicar porqué el modelo en rango [0,1] se comporta mejor que el modelo en rango [-1,1]. Para llevar a cabo este análisis se muestran las distribuciones de cada modelo, en lo que a número de fusiones se refiere.

Modelo en [-1,1]		Modelo en [0,1]	
Grupo de usuarios	Número fusiones	Grupo de usuarios	Número fusiones
1	371	1	2330
2	7	2	3219
3	426	3	2932
4	10285	4	367
5	11	5	575
6	17	6	450
7	12	7	88
8	21	8	70
9	12	9	86
10	10	10	4002
11	10	11	6649
12	14	12	91
13	630	13	128
14	20	14	63
15	9	15	265
16	9	16	61
17	8	17	24373
18	9	18	254
19	7	19	4025
20	414	20	77
21	86	21	3597
22	10	22	7110
23	10	23	421
24	7	24	6222
25	10	25	7690
26	11	26	736
27	13	27	88
28	363	28	562
29	30	29	210
30	11	30	97
31	8	31	2230
32	539	32	93
33	11	33	322
34	737	34	79
35	261	35	112
36	9	36	60
37	474	37	91
38	199	38	75
39	7	39	123
40	10	40	58
41	8	41	3067
42	10	42	85
43	9	43	279
44	11	44	2413
45	353	45	75
46	12	46	3879
47	431	47	148
48	12	48	69
49	8	49	83
50	1	50	1

Ilustración 33. Distribución modelos

Analizando los datos de la ilustración anterior que muestran para cada grupo de usuarios de ambos modelos, el número de fusiones que sufrió cada grupo, se puede observar que la distribución del modelo en [0,1] es mucho más uniforme que la distribución del modelo en [-1,1]. A su vez puede observarse que la suma del número de fusiones de todos los grupos de cada modelo no es cien mil que es el número de usuarios con que se entreno, por lo tanto, existen grupos que fueron eliminados del modelo, se remite al lector al apartado 6.2 del presente documento para comprender la eliminación de grupos de usuario del modelo. Por lo tanto, se puede concluir que existe pérdida de conocimiento en ambos modelos, no obstante, a continuación se analiza como de grande es la pérdida de conocimiento por eliminación de grupos de usuarios en ambos modelos.

Modelo en [-1,1]	Modelo en [0,1]
Fusiones grupos eliminados	Fusiones grupos eliminados
84037	9820

Ilustración 34. Perdida conocimiento

Como puede observarse la pérdida de conocimiento en el modelo [-1,1] es mucho mayor que en el modelo [0,1], esto se debe a que al ser peor la distribución de las fusiones en el modelo [-1,1], en este modelo se eliminan mas grupos, ya que las certezas de sus filas son más bajas y el modelo siempre elimina primero las filas que aportan menos información.

Para concluir con el análisis realizado, se deben mencionar los parámetros de ambos modelos, ya que estos intervinieron en los resultados de la experimentación. Destacar que limitar el número de grupos de la base de conocimiento del modelo a cincuenta restringe mucho y permite que el modelo en [0,1], al tener menor representación, pueda clasificar mejor que el modelo en [-1,1]. Sin embargo, se observa que si este número de grupos se aumentase, el funcionamiento del modelo en [-1,1] mejoraría y superaría al del modelo en [0,1]. Así pues, se concluye que los parámetros del sistema, como puedan ser el máximo de grupos de la base de conocimiento y el máximo número de filas de la base de conocimiento limitado a ocho mil han perjudicado al modelo en [-1,1] y favorecido al modelo en [0,1] lo que permite pues explicar los resultados obtenidos durante la fase de experimentación.

8. Conclusiones y líneas futuras

El proyecto que nos ocupa como todo proyecto de investigación supone un gran reto en lo que a modificaciones de la planificación inicial se refiere. Durante las distintas fases del proyecto, todas las planificaciones y especificaciones fueron modificadas una y otra vez con el fin de mejorar los encajes, las fusiones o incluso el rendimiento. Por lo tanto, la cantidad de problemas y obstáculos encontrados a lo largo del proyecto podría decirse que ha sido innumerable. No obstante, el resultado del proyecto se considera más que satisfactorio, pues tal y como se vio se lograron sendos modelos de usuario con una tasa de acierto en torno al 13% en un dominio muy difuso en el que él la tasa de acierto de los valores por defecto no supera el 1%. Así pues, resumiendo los datos obtenidos durante la experimentación del proyecto se obtuvieron tasas de acierto entre un 10% y un 13% en un dominio muy difuso. De igual forma se pudo comprobar porqué un modelo de usuarios realizando un tratamiento de datos en rango [0,1] se comporta mejor que uno realizando un tratamiento de datos en rango [-1,1] y deja una puerta abierta a una nueva experimentación en la que se ajusten los parámetros de ambos modelos cambiando el número máximo de grupos que se almacenan y el número máximo de filas de información que guarda el modelo. Así pues, se puede concluir que incrementando el número de grupos de usuario en el modelo, así como el número de filas de información del modelo, se obtendría una curva en la que el modelo en [-1,1] mejoraría y superaría al modelo en [0,1].

De esta manera se intuye que, si en lugar de trabajar con un dominio tan abierto y difuso, se trabajase con un dominio restringido los porcentajes de acierto de ambos modelos se incrementarían notoriamente. No obstante, esta experimentación con un dominio restringido se propone como línea futura de investigación aunque intuitivamente se puede afirmar que los resultados del modelo mejorarían en un dominio más cerrado.

Conviene también destacar el valor añadido que ofrece el proyecto a las líneas de investigación llevadas a cabo en este campo a lo largo de los años, si bien todas las líneas de investigación mantenidas hasta la fecha se centran en modelos de estereotipos, es decir, modelos creados para un dominio y aplicación concretos y que no son capaces de reajustarse en función del dominio, el trabajo propuesto en el presente documento ofrece un modelo de usuarios de grupos, es decir, un modelo basado en la experiencia y no en perfiles establecidos. Por tanto, el modelo propuesto es capaz de funcionar en distintos dominios y es totalmente independiente de la aplicación. Al tratarse, por tanto, de un modelo “automodelado”, éste nos permite aumentar el número de ranuras para grupos de usuarios y por lo tanto mejorar la clasificación que realiza el modelo.

De igual forma el proyecto aportó gran satisfacción personal durante su realización como a su finalización y gran valor formativo ya que permitió al autor desarrollar una investigación en torno a dos temas de gran interés: al modelado de usuarios y a la interacción natural. La realización del proyecto también permitió profundizar en otras áreas como la programación en Pl/Sql, la integración de Java con Oracle o la programación concurrente. Finalmente el proyecto tiene también el valor añadido de ser la base para una futura investigación doctoral.

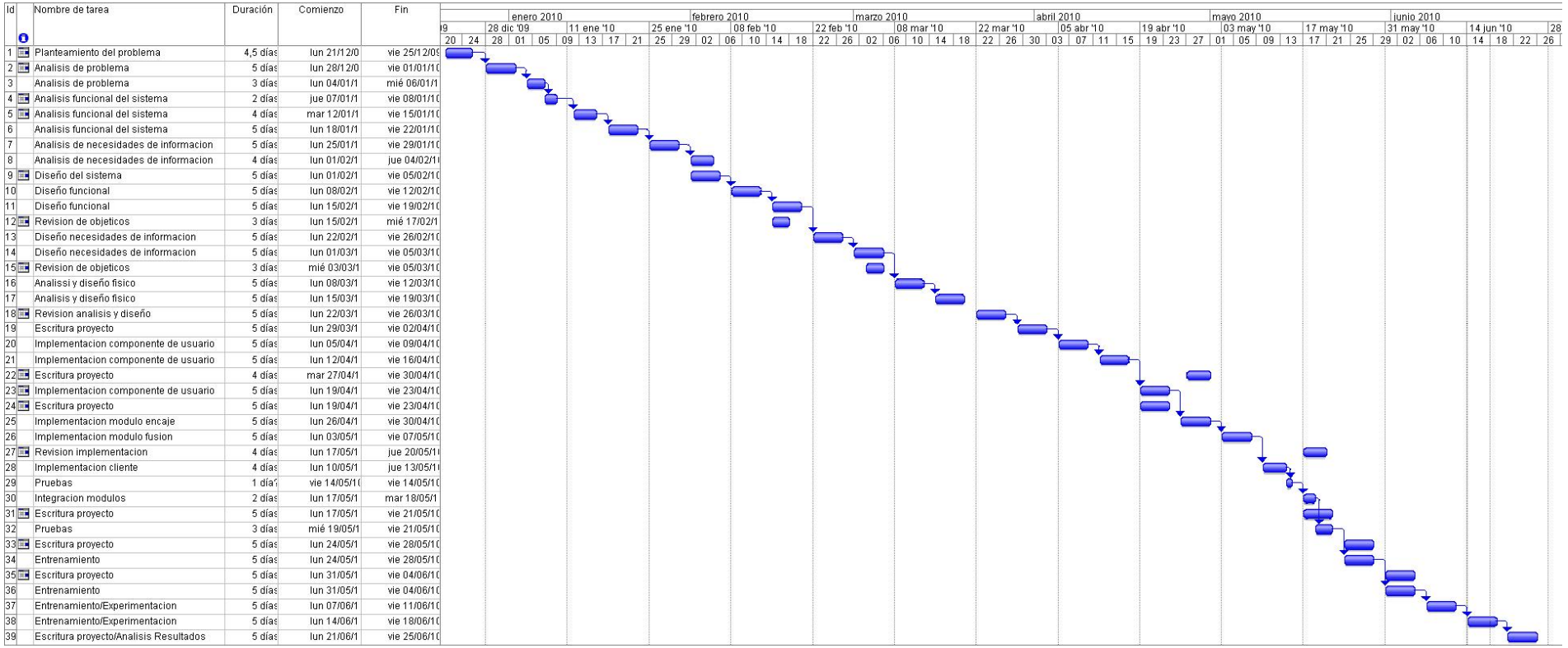
Por lo tanto, entre los valores añadidos del proyecto podemos destacar que actualmente se cuenta con un modelo de usuarios capaz de funcionar en cualquier dominio, que en un dominio difuso obtuvo unos resultados que si bien no son excelentes sí son muy superiores a la tasa de acierto de los valores por defecto del dominio. De igual forma la implementación del modelo permitió comparar distintos tratamientos de datos y evaluar cuál de ellos se comporta mejor, así como conocer las razones por las que un tratamiento de datos es mejor para un caso particular u otro. Conviene también destacar la gran cantidad de alternativas que la implementación del modelo ofrece.

Una posible ampliación al modelo implementado sería la introducción de una ontología que proporcionase al modelo conocimiento sobre las distancias entre ciertas características y permitiese discernir qué características son cercanas y cuáles lejanas. Sin duda esta ampliación dotaría al modelo de una capacidad semántica mucho mayor de la actual mejorando así el funcionamiento de éste.

Por otra parte, el modelo se encuadra dentro de un proyecto de interacción natural llamado “Interactor”, en el que el modelo de usuario propuesto debe interaccionar con un modelo emocional, un modelo de dialogo y/o un modelo de situación. La integración del sistema de interacción natural se propone como línea futura debido a la complejidad de algunos de los módulos del sistema, como por ejemplo el automodelo.

9. Presupuesto

El proyecto “Modelado de usuarios” se dividió en cuatro grandes fases tal y como se vio en el capítulo cuarto de este documento, las cuatro grandes fases del proyecto son, la creación del primer prototipo, una segunda fase de propuesta de alternativas, una tercera fase de trabajo con datos reales y finalmente el desarrollo del prototipo final y su evaluación. De igual forma y como todo proyecto que conlleva una fase de desarrollo, existen tareas de análisis, de diseño de implementación y una experimentación con los prototipos creados. El siguiente diagrama de Gantt expone la división en tareas del proyecto, así como la planificación de éste.



Id	Nombre de tarea	Duración	Comienzo	Fin
1	Planteamiento del problema	4,5 días	lun 21/12/0	vie 25/12/09
2	Analisis de problema	5 días	lun 28/12/0	vie 01/01/10
3	Analisis de problema	3 días	lun 04/01/1	mié 06/01/1
4	Analisis funcional del sistema	2 días	jue 07/01/1	vie 08/01/10
5	Analisis funcional del sistema	4 días	mar 12/01/1	vie 15/01/10
6	Analisis funcional del sistema	5 días	lun 18/01/1	vie 22/01/10
7	Analisis de necesidades de informacion	5 días	lun 25/01/1	vie 29/01/10
8	Analisis de necesidades de informacion	4 días	lun 01/02/1	jue 04/02/1
9	Diseño del sistema	5 días	lun 01/02/1	vie 05/02/10
10	Diseño funcional	5 días	lun 08/02/1	vie 12/02/10
11	Diseño funcional	5 días	lun 15/02/1	vie 19/02/10
12	Revision de objeticos	3 días	lun 15/02/1	mié 17/02/1
13	Diseño necesidades de informacion	5 días	lun 22/02/1	vie 26/02/10
14	Diseño necesidades de informacion	5 días	lun 01/03/1	vie 05/03/10
15	Revision de objeticos	3 días	mié 03/03/1	vie 05/03/10
16	Analissi y diseño fisico	5 días	lun 08/03/1	vie 12/03/10
17	Analisis y diseño fisico	5 días	lun 15/03/1	vie 19/03/10
18	Revision analisis y diseño	5 días	lun 22/03/1	vie 26/03/10
19	Escritura proyecto	5 días	lun 29/03/1	vie 02/04/10
20	Implementacion componente de usuario	5 días	lun 05/04/1	vie 09/04/10
21	Implementacion componente de usuario	5 días	lun 12/04/1	vie 16/04/10
22	Escritura proyecto	4 días	mar 27/04/1	vie 30/04/10
23	Implementacion componente de usuario	5 días	lun 19/04/1	vie 23/04/10
24	Escritura proyecto	5 días	lun 19/04/1	vie 23/04/10
25	Implementacion modulo encaje	5 días	lun 26/04/1	vie 30/04/10
26	Implementacion modulo fusion	5 días	lun 03/05/1	vie 07/05/10
27	Revision implementacion	4 días	lun 17/05/1	jue 20/05/1
28	Implementacion cliente	4 días	lun 10/05/1	jue 13/05/1
29	Pruebas	1 día?	vie 14/05/1	vie 14/05/10
30	Integracion modulos	2 días	lun 17/05/1	mar 18/05/1
31	Escritura proyecto	5 días	lun 17/05/1	vie 21/05/10
32	Pruebas	3 días	mié 19/05/1	vie 21/05/10
33	Escritura proyecto	5 días	lun 24/05/1	vie 28/05/10
34	Entrenamiento	5 días	lun 24/05/1	vie 28/05/10
35	Escritura proyecto	5 días	lun 31/05/1	vie 04/06/10
36	Entrenamiento	5 días	lun 31/05/1	vie 04/06/10
37	Entrenamiento/Experimentacion	5 días	lun 07/06/1	vie 11/06/10
38	Entrenamiento/Experimentacion	5 días	lun 14/06/1	vie 18/06/10
39	Escritura proyecto/Analisis Resultados	5 días	lun 21/06/1	vie 25/06/10

Ilustración 35. Diagrama de Gantt

Tras el diagrama de Gantt planificado para los seis meses de duración del proyecto con el detalle de las tareas y la planificación del proyecto, se incluye un presupuesto detallado de los costes del proyecto.

Cantidad	Descripción	Precio unitario hora	Precio unitario	Total
2	Jefes de proyecto (400 horas)	35	28.000,00 €	56.000,00 €
1	Analista desarrollador (900 horas)	25	22.500,00 €	22.500,00 €
1	Servidor de aplicaciones (amortizacion 20% anual)		400,00 €	400,00 €
1	Servidor de bases de datos (amortizacion 20% anual)		350 €	350,00 €
2	PC`s (compra)		270 €	540,00 €
1	Fungibles		300 €	300,00 €
			Subtotal: (o Base Imponible)	**78.900,00 €**
			Impuestos: (ejemplo: 16% IVA	12.624,00 €
			TOTAL:	**91.524,00 €**

Ilustración 36. Presupuesto

El presupuesto final de este proyecto asciende a una cantidad de 91.524 euros.

Leganés a 28 de Junio de 2010

El ingeniero proyectista

Fdo. Leonardo Castaño Zabaleta

Bibliografía

[Brajnik, G.& Tasso, C., 2002]. A shell for developing non-monotonic user modeling systems. Laboratorio di Intelligenza Artificiale, Dipartimento di Matematica e Informatica, Università di Udine, Via Zanon 6, 33100 Udine, Italy.

[Calle, F.J., 2005] Interacción Natural Mediante Procesamiento Intencional: Modelo de Hilos en diálogos. [PhD Thesis Universidad Politécnica de Madrid].

[del Coso, A., 2009]. Desarrollo de infraestructuras para el modelado de usuarios. Universidad Carlos III de Madrid. Disponible [Internet]: <http://earchivo.uc3m.es/bitstream/10016/8544/1/PFC_Ana_Coso_Santos.pdf> [18/06/2010].

[DRH Miller, T Leek & RM Schwartz, 1999]. A hidden Markov model information retrieval system. Berkeley, California, United States. pp. 214 – 221.

[Eclipse, 2010]. IDE multiplataforma. Eclipse Foundation. Disponible [Internet]: <http://www.eclipse.org/> [18/06/2010].

[Finin, T. W., 1989]. GUMS - A General User Modeling Shell. In: User Models in Dialog Systems. Springer Verlag, pp. 411-430.

[Horvitz, E., 1990]. Lumiere Project: Bayesian Reasoning for Automated Assistance. Disponible [Internet]: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.628&rep=rep1&type=pdf>.

[Java, 2010] Lenguaje de programación Java. Sun Microsystems. Disponible [Internet]: <http://es.sun.com/> [18/06/2010].

[Kay, J., 1995]. The sum Toolkit for reusable, long term user models. User Modeling and User-Adapted Interaction 4(3), 149-196.

[Kobsa, A., 1989]. A taxonomy of beliefs and goals for user models in dialog systems. In: A. Kobsa. and W. Wahlster (eds.), User Models in Dialog Systems. Springer-Verlag, Berlin, Heidelberg, pp. 52-68.

[Kobsa, A. & Pohl, W., 1995]. The user modelling shell system BGP-MS, in User Modelling and User-adapted interaction 4(2), pp. 59-106.

[Konstan, J. A., et. al., 1997]. GroupLens: Applying collaborative filtering to Usenet news. Communications of the ACM. 40(3). pp. 77 – 87.

[Oracle 11G, 2010]. Sistema gestor de bases de datos. Oracle España. Disponible [Internet]: <http://www.oracle.com/global/es/index.html> [18/06/2010].

[Orwant, J., 1994]. Heterogeneous learning in the Doppelgänger user modeling system. The Media Laboratory, MIT, 20 Ames St., 02139 Cambridge, MA, USA.

[Paiva, A.& Self, J., 1995]. TAGUS — A user and learner modeling workbench. Department of Computing, Lancaster University, LAI 4YR Lancaster, UK.

[Pl/Sql, 2010]. Lenguaje de programación Pl/Sql. Oracle España. Disponible [Internet]: <http://www.oracle.com/technology/tech/pl_sql/index.html> [18/06/2010].

[Sherman, E. H. & Shortliffe, E. H., 1993]. A User-Adaptable Interface to Predict Users' Needs. In: Adaptive User Interfaces: Principles and Practise. Amsterdam. North Holland Elsevier.

[SQL Developer, 2010]. Cliente desarrollo bases de datos. Oracle España. Disponible [Internet] : < http://www.oracle.com/technology/products/database/sql_developer/index.html> [18/06/2010].

[SSL, 2010]. Secure socket layer. Netscape Communications Corporation. Disponible [Internet]: <http://netscape.aol.com/> [18/06/2010].

[Vergara, H., 1994]. PROTUM: A Prolog Based Tool for User Modeling. WIS Memo 10, WG Knowledge-Based Information Systems, Department of Information Science, University of Konstanz, Germany.

[Zukerman, D.W. & Albrecht, I., 2001]. Predictive Statistical Models for User Modeling. School of Computer Science and Software Engineering, Monash University, Clayton, Victoria, 3800, Australia.